最全面的
OpenClaw中文教程

第一部分：零基础入门（3章）

第1章：认识OpenClaw:是什么？能帮你做什么？

第2章：环境搭建：5分钟完成部署（多种方案任选）

第3章：发送第一条消息，配置你的专属AI助手**第二部**

分：4大核心功能（4章）

第4章：本地文件管理神器（效率提升81%）

第5章：个人知识库（网页/论文/GitHub一键存档）

第6章：日程管理（微信截图秒变日历事件）

第7章：自动化工作流（定时任务/网站监控/AI日报）

第三部分：进阶技能（4章）

第8章：Skills扩展（1715个技能让AI无所不能）

第9章：多平台集成（飞书/企微/钉钉/QQ一键接入）

第10章：API服务集成（绘图/Notion/视频/语音）

第11章：高级配置（多模型切换/成本优化/性能调优）

第四部分：实战案例（4章）

第12章：5类人群的效率提升实战

第13章：高级自动化工作流（多Skills组合/知识图谱）

第14章：创意应用探索（AI绘画/视频/翻译/数据分析）

第15章：超级个体实战案例（一人公司/自由职业）

第1章：认识OpenClaw

> 在开始使用 OpenClaw之前，先了解它是什么，为什么要用它，以及它能带来什么。

> ⚠️ **版本说明**：由于 OpenClaw 仍在快速开发中，本教程基于 **2026.2.9** 版本编写。该版本经过充分验证，稳定可靠。

> 💡 **重要前提**：OpenClaw 预装了 **49个内置技能（Skills）**，本教程中的大部分功能演示都基于这些内置技能。这些技能涵盖文件管理、知识管理、日程管理、自动化等核心场景，开箱即用。如果某些功能无法使用，请先确认相关技能是否已启用（详见第8章：Skills扩展）。

1.1 什么是OpenClaw

一句话介绍

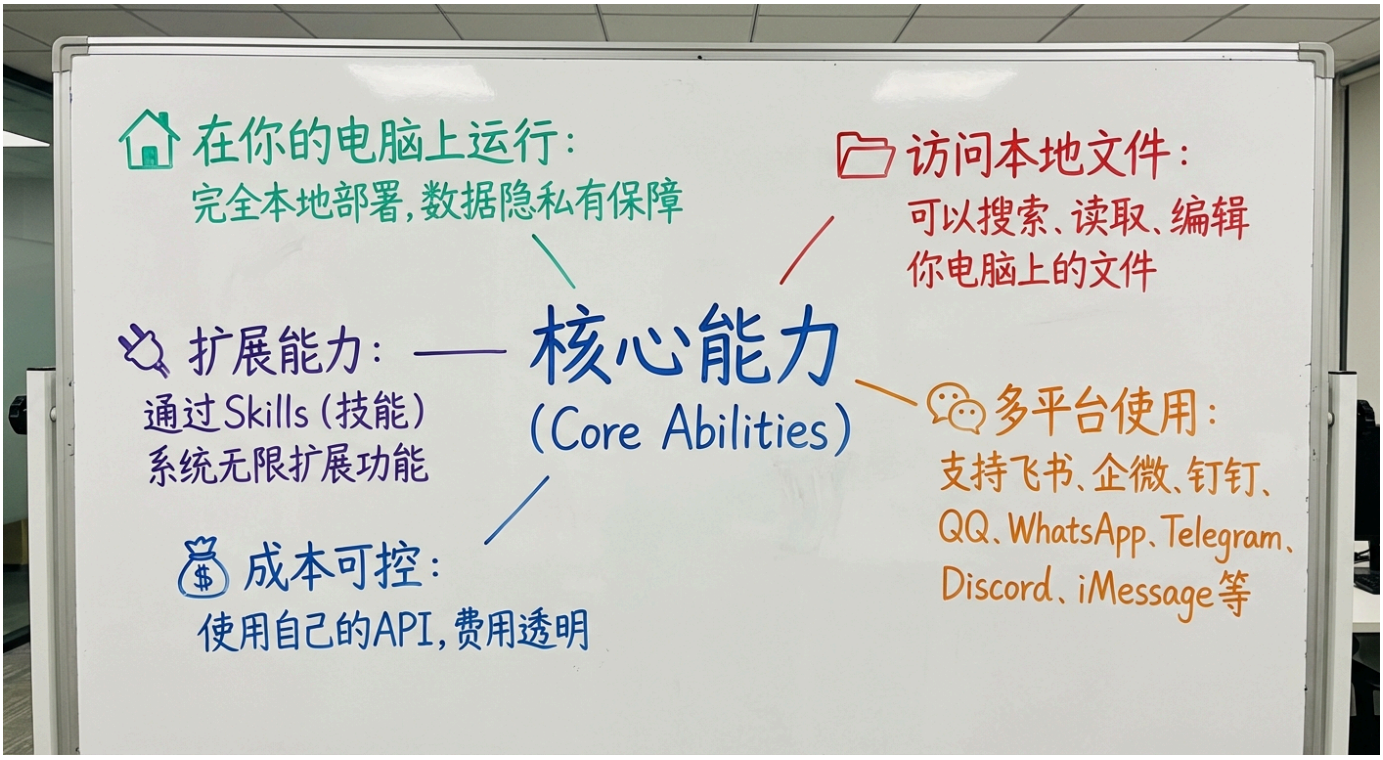
OpenClaw 是一个**开源的 AI 智能体 Gateway 网关**，让你可以在本地部署 AI 助手，访问本地文件，通过多个平台（飞书、企微、QQ等）随时使用。

> **项目名称演变**：OpenClaw 原名 Clawdbot，因 Anthropic 商标顾虑在 2026年1月27日更名为 Moltbot（过渡名），最终在1月30日确定为 OpenClaw。三个名字本质是同一个项目，功能完全一致。

核心能力

- 🏠 **本地部署**：在你的电脑上运行，数据隐私有保障
- 📁 **文件访问**：可以搜索、读取、编辑你电脑上的文件
- 🔌 **无限扩展**：通过 Skills（技能）系统扩展功能
- 💬 **多平台使用**：支持飞书、企微、钉钉、QQ、Telegram、Discord等
- 💰 **成本可控**：使用自己的 API，费用透明

工作原理

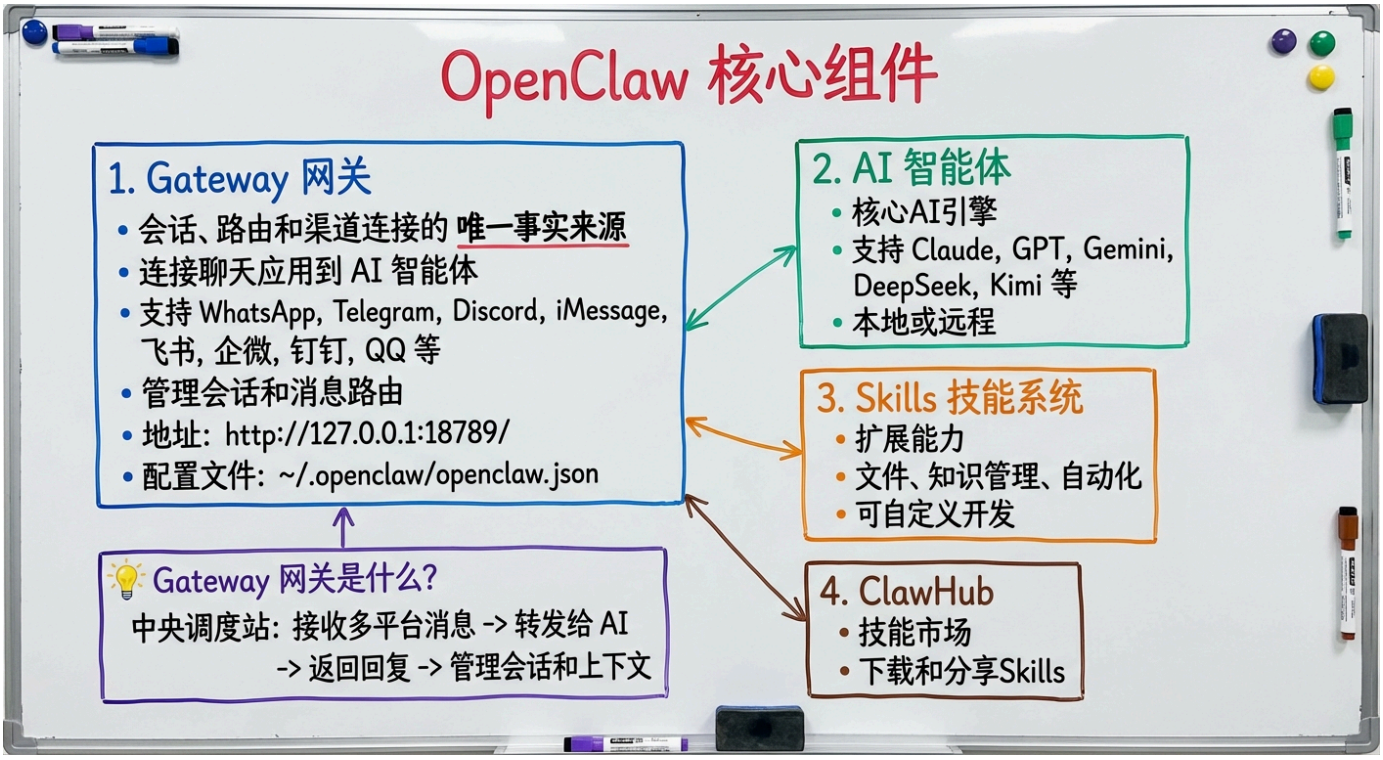


OpenClaw 通过 Gateway 网关将聊天应用连接到 AI 智能体。Gateway 是会话、路由和渠道连接的**唯一事实来源**。

核心组件：

1. **Gateway 网关**
   - 连接各个聊天平台（飞书、企微、QQ、Telegram等）
   - 管理会话和消息路由
   - 默认地址：`http://127.0.0.1:18789/`
   - 配置文件：`~/.openclaw/openclaw.json`
2. **AI 智能体**
   - 支持 Claude、GPT、Gemini、DeepSeek、Kimi 等多种模型
   - 可以本地运行或远程调用
3. **Skills 技能系统**
   - 文件管理、知识管理、自动化等
   - 可自定义开发
4. **ClawHub**
   - 技能市场，可以下载和分享 Skills

> 💡 **Gateway 网关是什么?**
> Gateway 是 OpenClaw 的核心服务，它像一个"中央调度站"，负责：
>
> - 接收来自不同平台的消息（飞书、企微、QQ、Telegram等）
> - 将消息转发给 AI 智能体处理
> - 把 AI 的回复发送回对应平台
> - 管理所有的会话和上下文
>
> 这就是为什么你可以在任何平台上使用 OpenClaw，因为 Gateway 统一管理了所有的连接。



1.2 为什么选择OpenClaw

与在线AI的本质区别

OpenClaw 与传统在线 AI 服务在部署方式、数据隐私和功能扩展等方面存在本质差异，如表 1-3 所示。

表 1-3 OpenClaw 与在线 AI 对比

特性	OpenClaw	ChatGPT/Claude网页版
部署方式	本地/云端	在线服务
数据隐私	✅ 完全掌控	⚠️ 上传到服务器
本地文件访问	✅ 支持	❌ 不支持
系统操作	✅ 支持	❌ 不支持
功能扩展	✅ Skills系统	❌ 固定功能
成本	按需付费	订阅制
多平台集成	✅ 支持	⚠️ 有限

简单来说：

- ChatGPT/Claude：像在网吧上网，方便但受限
- OpenClaw：像自己的电脑，自由但需要配置

五大核心优势

1. 本地部署，保护隐私

场景：你需要AI帮你整理公司的财务报表

- ❌ **在线AI**：需要上传文件到服务器，存在泄密风险
- ✅ **OpenClaw**：文件不离开你的电脑，完全安全

2. 访问本地文件和系统

场景：你想找一张去年的发票

- ❌ **在线AI**：无法访问你的电脑，只能手动找
- ✅ **OpenClaw**：直接搜索电脑上的所有文件，秒找到

真实案例：

```
我：帮我找一下我电脑上的一张发票，里面详情是买了一个跑步机
OpenClaw：[搜索中...] 找到了！这是您的跑步机发票 [发送文件]
```

3. 可扩展的Skills生态

场景：你想让AI帮你画图

- ❌ **在线AI**：功能固定，不支持就没办法
- ✅ **OpenClaw**：安装 Banana Skills，立即支持画图

可用的Skills：

- 文件管理：智能搜索、批量处理、自动整理
- 知识管理：网页剪藏、笔记同步、论文管理
- 日程管理：日历同步、智能提醒
- 自动化：定时任务、网站监控、日报推送
- 工具类：截图、翻译、画图、视频生成

4. 多平台支持

场景：你在外面，想让AI帮你处理文件

- ❌ **在线AI**：需要打开网页，操作不便
- ✅ **OpenClaw**：打开飞书/企微/QQ，直接发消息

支持的平台：

- 🏢 **企业微信**：团队协作
- 📱 **钉钉**：办公自动化
- 🚀 **飞书**：项目管理
- 💬 **QQ**：个人助手
- 🌐 **Telegram/Discord**：国际平台

5. 成本可控

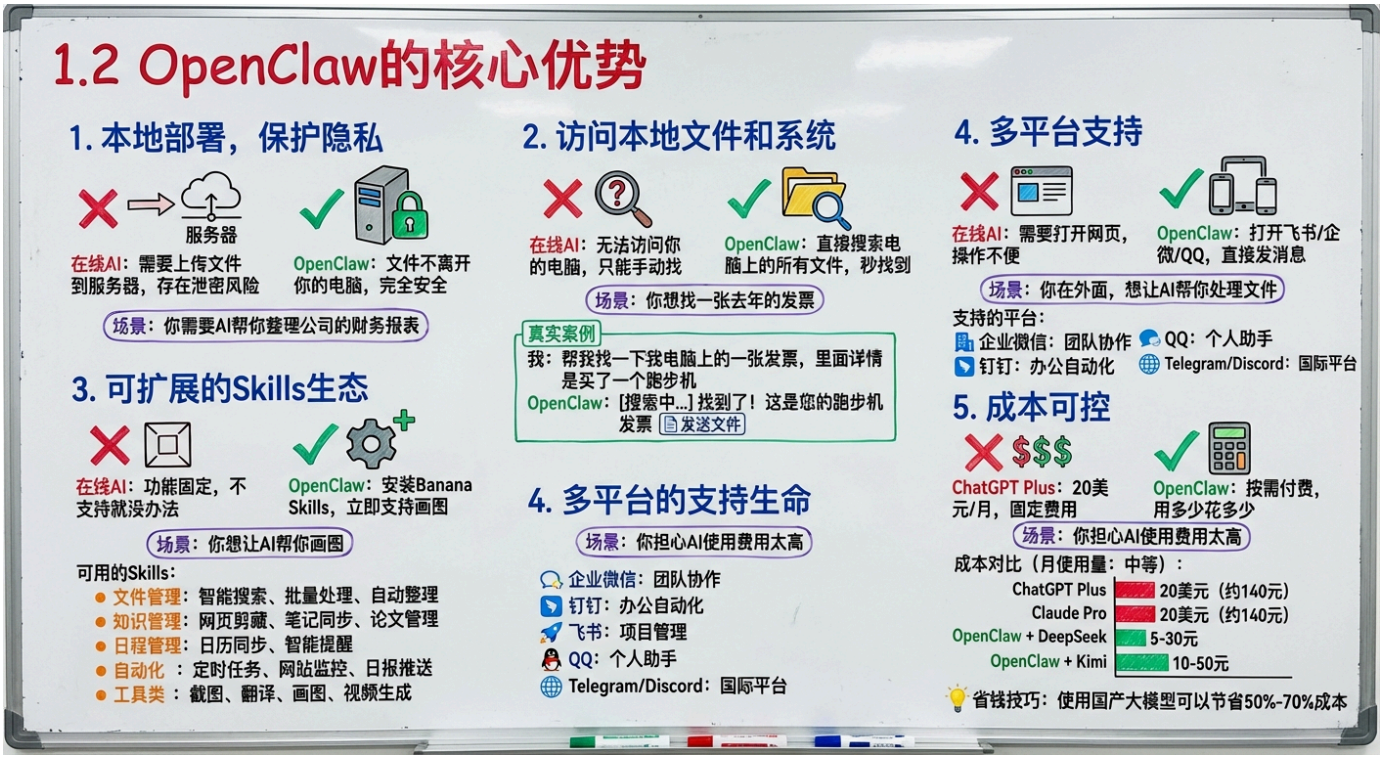
场景：你担心AI使用费用太高

- ❌ **ChatGPT Plus**：20美元/月，固定费用
- ✅ **OpenClaw**：按需付费，用多少花多少

成本对比（月使用量：中等）：

- ChatGPT Plus：20美元（约140元）
- Claude Pro：20美元（约140元）
- OpenClaw + DeepSeek：5-30元
- OpenClaw + Kimi：10-50元

💡 **省钱技巧**：使用国产大模型可以节省50%-70%成本



1.3 OpenClaw vs 主流AI工具

OpenClaw 与主流 AI 工具在功能定位和成本结构上存在显著差异。下面从核心功能、成本和适用人群三个维度进行详细对比。

核心功能对比

OpenClaw 在本地文件访问、系统操作和自动化任务方面具有明显优势，如表 1-1 所示。

表 1-1 主流 AI 工具功能对比

功能特性	OpenClaw	ChatGPT Plus	Cursor	Claude Pro
本地文件访问	✅ 完整支持	❌ 不支持	✅ 支持	❌ 不支持
系统操作	✅ 日历/备忘录/截图	❌ 不支持	⚠️ 有限	❌ 不支持
多平台集成	✅ 飞书/企微/钉钉/QQ	⚠️ 有限	❌ 不支持	⚠️ 有限
Skills扩展	✅ 1715+	❌ 不支持	⚠️ 插件	❌ 不支持
多模型切换	✅ 随意切换	❌ 固定GPT	✅ 支持	❌ 固定Claude
代码能力	✅ 强	⚠️ 中等	✅ 强	✅ 强
自动化任务	✅ 完整支持	❌ 不支持	❌ 不支持	❌ 不支持

成本对比

OpenClaw 采用按量付费模式，相比固定月费的在线服务可节省 73%-96% 的成本，如表 1-2 所示。

表 1-2 年度成本对比

项目	OpenClaw	ChatGPT Plus	Cursor	Claude Pro
月费	5-50元	140元	140元	140元
年费	60-600元	1680元	1680元	1680元
节省	-	节省73%-96%	节省73%-96%	节省73%-96%
计费方式	按量付费	固定月费	固定月费	固定月费

成本说明：

- OpenClaw 成本 = 服务器费用（可选）+ API 费用
- 本地部署：0元服务器 + 5-30元 API = 5-30元/月
- 云端部署：20元服务器 + 5-30元 API = 25-50元/月

适合人群

✅ **强烈推荐OpenClaw的人群：**

1. **超级个体/自由职业者** - 需要一个人顶一个团队
2. **知识工作者** - 需要管理大量文档和知识
3. **程序员** - 需要代码辅助和自动化
4. **内容创作者** - 需要素材管理和多平台发布
5. **注重隐私** - 不想上传敏感文件到云端
6. **成本敏感** - 想要高性价比的AI助手

⚠️ **如果你只需要简单对话，可以考虑：**

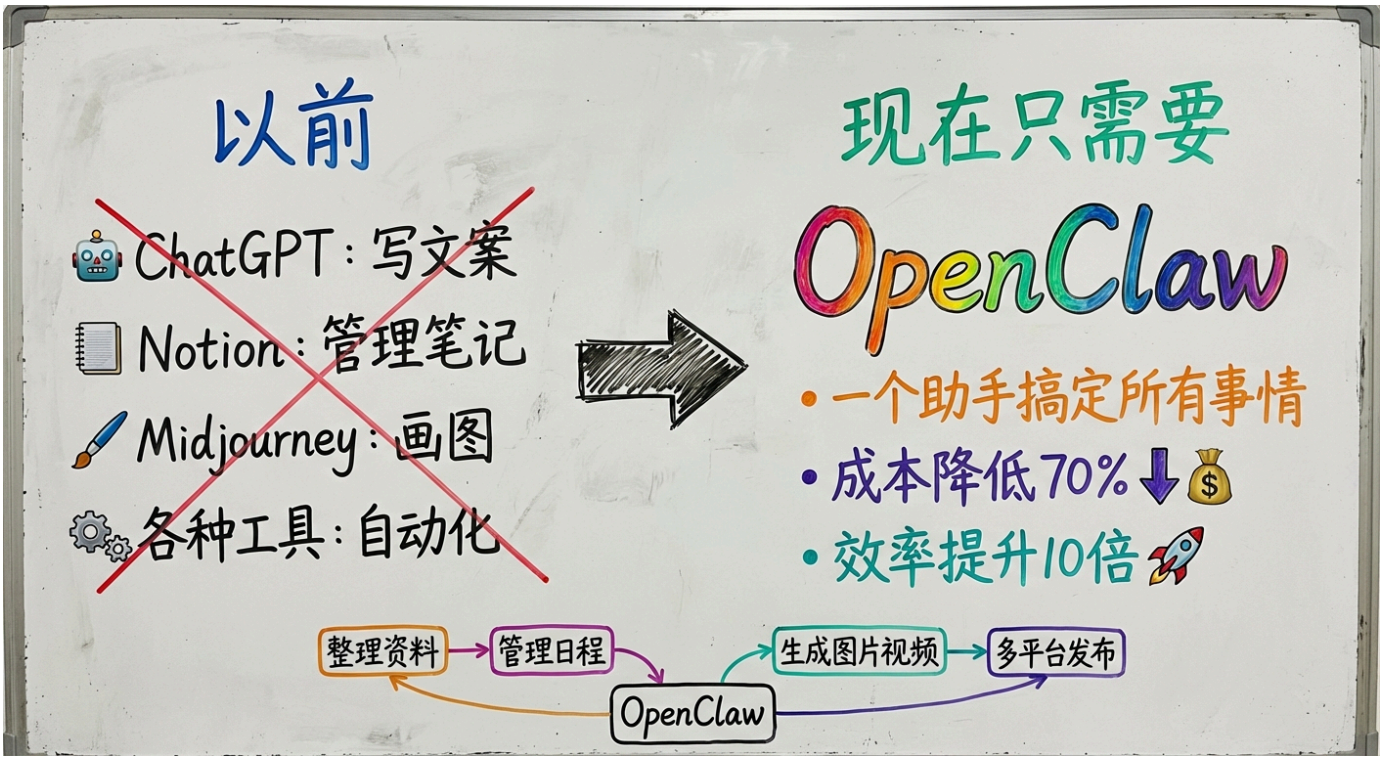
- ChatGPT Plus：最简单，开箱即用
- Claude Pro：长文本处理能力强

💡 **最佳方案：**

- 日常对话：ChatGPT/Claude
- 工作助手：OpenClaw（本地文件、自动化、多平台）
- 代码编写：Cursor + OpenClaw

真实案例：为什么我选择OpenClaw

> 我是一个内容创作者，每天需要：
>
> - 整理大量的资料和文章
> - 管理日程和任务
> - 生成图片和视频
> - 多平台发布内容
>
> 以前我需要：
>
> - ChatGPT：写文案
> - Notion：管理笔记
> - Midjourney：画图
> - 各种工具：自动化
>
> 现在只需要OpenClaw：
>
> - 一个助手搞定所有事情
> - 成本降低70%
> - 效率提升10倍



1.4 适用场景

✅ 适合使用 OpenClaw的场景

1. 个人效率提升

知识工作者：

- 整理大量文档和资料
- 管理日程和任务
- 自动化重复工作

程序员：

- 代码搜索和管理
- 技术文档整理
- 开发环境管理

内容创作者：

- 资料收集和整理
- 内容创作辅助
- 多平台发布

学生：

- 课程资料管理
- 论文阅读和笔记
- 学习计划管理

2. 知识管理

- 网页文章存档
- GitHub项目管理
- 论文笔记整理
- 跨设备同步

3. 编程开发

- 代码搜索和理解
- 技术文档查询
- 开发环境配置
- 项目管理

4. 团队协作

- 项目管理自动化
- 文档协作优化
- 会议记录整理
- 团队知识库建设

5. 创意工作

- AI绘画工作流

- 视频脚本生成
- 多语言翻译
- 数据分析自动化

❌ 不适合使用 OpenClaw的场景

1. 纯在线对话

如果你只是想偶尔和AI对话，不需要访问本地文件：

- ChatGPT网页版更方便
- 无需安装配置
- 打开就能用

2. 专业代码编辑

如果你只需要代码补全和编辑：

- Cursor更专业
- GitHub Copilot更方便
- 集成度更高

3. 移动端为主

如果你主要在手机上使用：

- ChatGPT App更方便
- Claude App更流畅
- 无需配置

4. 不想折腾

如果你：

- 不想学习新工具
- 不想配置环境
- 只想开箱即用

那么在线AI服务更适合你。

💡 最佳实践

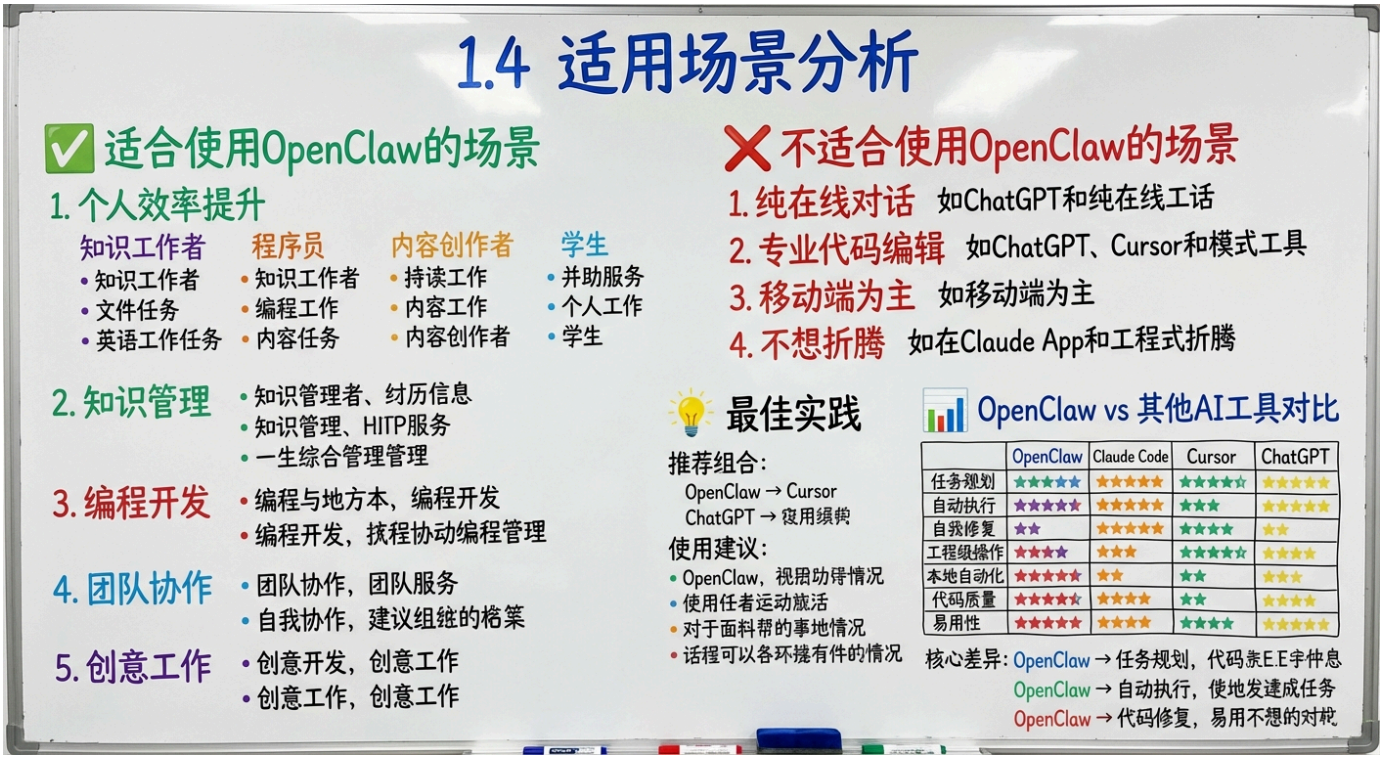
推荐组合：

1. **OpenClaw**：日常工作助手
2. **Cursor**：代码编辑
3. **ChatGPT**：移动端对话

使用建议：

- 在电脑上：优先使用 OpenClaw

- 写代码时：使用Cursor
- 在外面时：使用ChatGPT App



1.5 OpenClaw vs 其他AI工具能力对比

不同 AI 工具在任务规划、自动执行、代码质量等维度上各有侧重。OpenClaw 在任务规划和自动化方面表现突出，如表 1-4 所示。

表 1-4 AI 工具能力维度对比

能力维度	OpenClaw	Claude Code	Cursor	ChatGPT
任务规划	⭐⭐⭐⭐⭐	⭐⭐⭐	⭐⭐⭐	⭐⭐⭐
自动执行	⭐⭐⭐⭐⭐	⭐⭐⭐	⭐⭐⭐	⭐⭐
自我修复	⭐⭐⭐⭐⭐	⭐⭐	⭐⭐	⭐
工程级操作	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐
本地自动化	⭐⭐⭐⭐⭐	⭐⭐	⭐⭐	⭐
代码质量	⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐
易用性	⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐

核心差异：

- **OpenClaw**：强在任务规划和自动执行完整工程流程
- **Claude Code/Cursor**：强在代码质量与理解
- **ChatGPT**：强在对话体验和易用性

本章小结

通过本章，你应该了解了：

✅ **OpenClaw是什么**：本地AI助手，通过Gateway连接多个平台
✅ **核心优势**：本地部署、文件访问、可扩展、多平台、成本低
✅ **与其他AI的区别**：更适合作为个人工作助手
✅ **适用场景**：效率提升、知识管理、编程开发、团队协作
✅ **不适合场景**：纯对话、移动端为主、不想折腾

思考题

1. 你目前使用什么AI工具？遇到了什么问题？
2. OpenClaw的哪个功能最吸引你？
3. 你打算用OpenClaw做什么？

下一章：第2章：环境搭建 - 5分钟完成安装

返回目录：README

第2章：环境搭建

本章将手把手教你安装 OpenClaw。

📋 前提条件与推荐配置

推荐配置

为了获得最佳体验，我们推荐：

操作系统：

- **Mac（强烈推荐）**：原生支持最完善，可操作日历、备忘录、截图等系统功能
- Windows：完全可用，但部分系统集成功能受限
- Linux：适合开发者，配置灵活

IM工具选择：

- 🌍 **国外用户**：推荐 **Telegram**（适配度最好，功能最完整）
- 🇨🇳 **国内用户**：推荐 **飞书**（现代化、开发友好、功能丰富）
- 备选：企业微信、钉钉、QQ

部署方式：

- 💻 **有Mac电脑**：推荐本地部署（体验最好，功能最全）
- ☁️ **无Mac或想24小时运行**：推荐云端部署（成本低，稳定可靠）

为什么推荐Mac？

OpenClaw在Mac上体验最好，因为：

- 原生支持最完善，系统集成度高
- 可以操作Mac日历、备忘录、提醒事项
- 截图功能完美支持
- 与iPhone、iPad无缝同步
- 文件管理更智能
- 开发环境配置简单

为什么推荐飞书（国内）？

- 现代化设计，用户体验好
- 开发者友好，API完善
- 支持富文本、文档、表格
- 消息推送稳定
- 免费版功能丰富

为什么推荐Telegram（国外）？

- 全球用户基础大
- API最完善，功能最强
- 支持Bot功能丰富
- 消息推送实时
- 隐私保护好

快速导航

推荐路径：

- **有Mac** → Mac本地部署 + 飞书配置
- ☁️ **无Mac/想24小时运行** → 云端一键部署 + 飞书配置

所有部署方式：

- 🖥 Mac本地部署（推荐）
- Windows本地部署
- Linux本地部署
- 云端一键部署
- 国内一键安装
- ☁️ Cloudflare Workers 部署（进阶）
- Docker 部署（可选）

配置指南：

- API配置指南
- 版本升级指南
- ? 常见问题解决

Mac本地部署（推荐）

> **最佳体验**：如果你有Mac电脑，强烈推荐本地部署，体验最好、功能最全！

为什么选择Mac本地部署？

优势：

- **系统集成**：可操作日历、备忘录、文件系统
- **隐私安全**：数据完全本地，不上传云端
- **响应速度快**：本地运行，无网络延迟
- **功能最全**：支持所有高级功能
- **成本低**：无需购买云服务器
- **开发友好**：方便调试和自定义

适合人群：

- 有Mac电脑的用户
- 注重隐私的用户
- 需要系统集成功能的用户
- 开发者和技术爱好者

系统要求

硬件要求：

- CPU：M系列芯片或Intel i5以上
- 内存：8GB以上（推荐16GB）
- 硬盘：10GB以上空闲空间

系统版本：

- macOS 12 Monterey 或更高版本
- 推荐 macOS 14 Sonoma 或 macOS 15 Sequoia

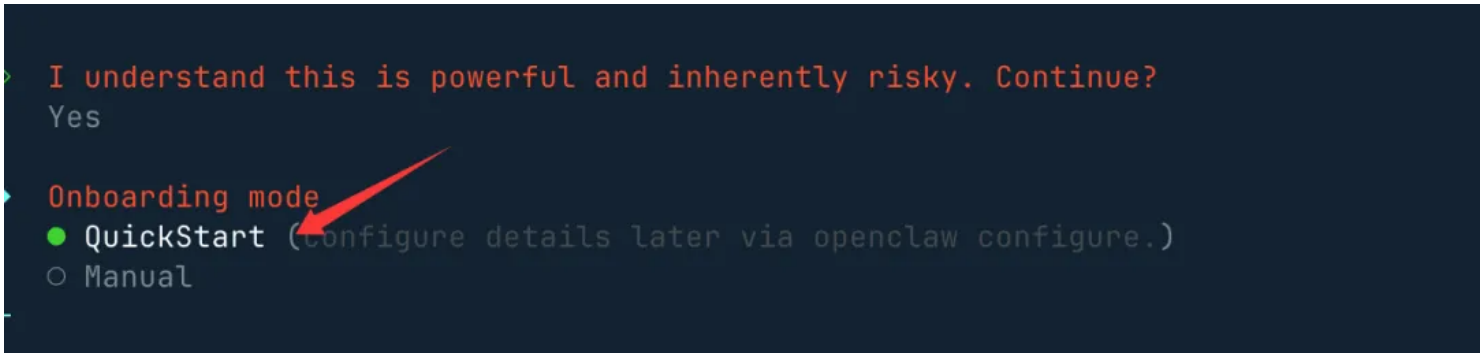
前置软件：

- Node.js 22.0.0+（会自动安装）
- Homebrew（可选，用于安装依赖）

安装步骤

第一步：打开终端

1. 按 `Command + 空格` 打开 Spotlight
2. 输入 `Terminal` 或 `终端`
3. 按回车打开终端



第二步：安装 OpenClaw

在终端中执行以下命令：

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

安装过程会自动：

- 检测系统环境
- 安装Node.js（如果未安装）
- 下载OpenClaw
- 配置环境变量

预计时间：2-5分钟

第三步：验证安装

安装完成后，执行以下命令验证：

```
openclaw --version
```

如果显示版本号（如 `2026.2.9`），说明安装成功！

第四步：初始化配置

运行配置向导：

```
openclaw onboard
```

配置流程：

1. 接受风险提示：

选择 `Yes` 继续

```
I understand this is powerful and inherently risky. Continue?
Yes

Onboarding mode
● QuickStart (Configure details later via openclaw configure.)
○ Manual
```

2. 选择启动模式：

推荐选择 `QuickStart` 快速启动：

```
I understand this is powerful and inherently risky. Continue?
Yes

Onboarding mode
● QuickStart (Configure details later via openclaw configure.)
○ Manual
```

3. 选择AI模型：

选择你的AI供应商（支持国内外主流模型）：

```
Model/auth provider
○ OpenAI
○ Anthropic
○ MiniMax
● Moonshot AI (Kimi K2 + Kimi Coding)
○ Google
○ OpenRouter
○ Qwen
○ Z.AI (GLM 4.7)
○ Copilot
○ Vercel AI Gateway
○ OpenCode Zen
○ Xiaomi
○ Synthetic
○ Venice AI
○ Skip for now
```

国内推荐：

- **Kimi（Moonshot AI）**：长文本专家，200万字上下文
- **DeepSeek**：性价比之王，推理能力强
- **智谱GLM**：中文理解好，多模态支持

4. 输入API Key：

根据选择的模型，输入对应的API Key（参见[API配置指南]）

5. 选择聊天工具：

- 如果要接入飞书/Telegram，选择对应选项
- 如果暂时不接入，选择 `None`（后续可配置）

```
Select channel (QuickStart)
○ Telegram (Bot API)
○ WhatsApp (QR link)
○ Discord (Bot API)
○ Google Chat (Chat API)
○ Slack (Socket Mode)
○ Signal (signal-cli)
○ iMessage (imsg)
○ Nostr (NIP-04 DMs)
○ Microsoft Teams (Bot Framework)
○ Mattermost (plugin)
○ Nextcloud Talk (self-hosted)
○ Matrix (plugin)
○ BlueBubbles (macOS app)
○ LINE (Messaging API)
○ Zalo (Bot API)
○ Zalo (Personal Account)
○ Tlon (Urbit)
● Skip for now (You can add channels later via `openclaw channels add`)
```

6. Gateway端口设置：

默认 `18789` 即可：

```
Configure skills now? (recommended)
Yes

Preferred node manager for skill installs
● npm
○ pnpm
○ bun
```

7. 选择Skills：

使用空格键选择你需要的技能，也可以直接跳过：

```
npm

Install missing skill dependencies
□ Skip for now (Continue without installing dependencies)
□ 🔐 1password
□ 📝 apple-notes
□ ⏰ apple-reminders
```

8. API Key配置：

没有的可以选择 `no` 跳过：

```
Install missing skill dependencies
Skip for now

Set GOOGLE_PLACES_API_KEY for goplaces?
No

Set GOOGLE_PLACES_API_KEY for local-places?
No
```

9. 启用Hooks：

推荐启用这三个钩子（用于内容引导、日志和会话记录）：

```
Enable hooks?
◻ Skip for now
◻ 🚀 boot-md
◻ 📝 command-logger
◻ 💾 session-memory
```

10. 完成配置：

配置完成后，会自动启动Gateway服务并打开Web UI（ `http://127.0.0.1:18789/chat` ）

第五步：验证安装

```
# 检查Gateway状态
openclaw channels status

# 应该显示：
# Gateway reachable.
```

日常使用

启动OpenClaw：

```
# 启动Gateway服务
openclaw gateway start

# 或使用systemd（推荐，开机自启）
openclaw gateway enable
```

访问Web UI：

打开浏览器访问： `http://127.0.0.1:18789/chat`

停止服务：

```
openclaw gateway stop
```

接入飞书（推荐）

Mac本地部署后，强烈推荐接入飞书，获得最佳体验：

1. 参考 [第9章：飞书Bot配置]
2. 配置完成后，可以在飞书中随时与OpenClaw对话
3. 支持文本、图片、文件等多种消息类型

常见问题

Q1：安装时提示权限不足？

```
# 使用sudo安装
curl -fsSL https://openclaw.ai/install.sh | sudo bash
```

Q2：如何更新OpenClaw？

```
openclaw update
```

Q3：如何卸载？

```
openclaw uninstall
```

Windows本地部署

> **Windows用户**：完全可用，但部分系统集成功能受限。

系统要求

硬件要求：

- CPU：2核以上
- 内存：4GB以上（推荐8GB）
- 硬盘：10GB以上空闲空间

操作系统：

- Windows 10 或 Windows 11

前置软件：

- Node.js 22.0.0+

部署方式选择

Windows有两种部署方式：

1. **WSL2 + Ubuntu（强烈推荐）**：官方推荐方式，提供完整Linux环境支持
2. **PowerShell原生部署**：纯Windows环境，适合不想使用WSL2的用户

方式一：WSL2 + Ubuntu部署（强烈推荐）

这是官方推荐的Windows部署方式，提供最完整的Linux环境支持。

第一步：启用WSL2

以管理员身份打开PowerShell，执行：

```
# 启用WSL功能
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /noresta
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

# 设置WSL 2为默认版本
wsl --set-default-version 2
```

重启计算机。

第二步：安装Ubuntu

方法一：Microsoft Store安装（推荐）

1. 打开Microsoft Store
2. 搜索「Ubuntu 22.04 LTS」或「Ubuntu 24.04 LTS」
3. 点击「获取」并安装
4. 首次启动设置用户名和密码

安装完成后会自动打开Ubuntu终端，按提示设置用户名和密码。

第三步：更新Ubuntu系统

在Ubuntu终端中执行：

```
# 更新软件包列表
sudo apt update && sudo apt upgrade -y

# 安装基础工具
sudo apt install -y curl git wget build-essential
```

第四步：安装Node.js 22+

```
# 添加NodeSource仓库
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -

# 安装Node.js
sudo apt install -y nodejs

# 验证版本（必须≥22.x）
node -v
npm -v
```

第五步：安装 OpenClaw

方法A：一键脚本安装

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

第六步：验证安装

```
# 查看版本
openclaw --version

# 查看帮助
openclaw --help

# 查看系统状态
openclaw status
```

第七步：配置Windows访问WSL2服务

由于OpenClaw运行在WSL2中，需要配置端口转发以便Windows访问。

创建启动脚本 `start-openclaw.bat` ：

```
@echo off
echo Starting OpenClaw Gateway in WSL2...
wsl -d Ubuntu-22.04 -u root service openclaw start
timeout /t 3
start http://localhost:18789
```

或直接在WSL2中启动：

```
# 在WSL2 Ubuntu终端中
openclaw gateway run --port 18789
```

然后在Windows浏览器访问 `http://localhost:18789`

方式二：PowerShell原生部署

适合不想使用WSL2的纯Windows用户。

第一步：安装Node.js 22+

方法一：官网下载安装

1. 访问 https://nodejs.org/zh-cn
2. 下载Windows安装包（LTS版本22.x）

3. 运行安装程序，勾选「自动安装必要的工具」

第二步：验证Node.js安装

```
# 打开PowerShell
node -v
npm -v
```

第三步：以管理员身份安装 OpenClaw

重要：必须以**管理员身份**运行PowerShell。

```
# 安装最新稳定版
npm install -g openclaw@latest

# 或安装汉化版
npm install -g @qingchencloud/openclaw-zh@latest
```

第四步：解决安装权限问题

如果遇到权限错误：

```
# 方法A：启用PowerShell脚本执行
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

# 方法B：修改npm安装目录
npm config set prefix "C:\npm"
npm config set cache "C:\npm-cache"

# 将目录添加到PATH
[Environment]::SetEnvironmentVariable("Path", $env:Path + ";C:\npm", "User")
```

第五步：验证安装

```
openclaw --version
openclaw --help
```

第六步：解决常见问题

问题：sharp模块加载失败

```
# 清理npm缓存
npm cache clean --force

# 重新安装
npm install -g openclaw@latest --force
```

问题：Windows Defender阻止

将OpenClaw安装目录添加到Windows Defender排除项：

```
C:\Users\你的用户名\AppData\Roaming\npm
C:\Users\你的用户名\.openclaw
```

初始化配置

安装完成后，需要运行初始化向导。

启动初始化向导

```
openclaw onboard --install-daemon
```

配置AI模型提供商

OpenClaw需要对接AI模型才能工作。

以配置Anthropic Claude（推荐）为例：

> **详细说明**: 完整的 API Key 配置方式和优先级说明请参考 [API Key 配置完整指南]

```
# WSL2或PowerShell
openclaw models auth add
# 按提示选择 anthropic
# 输入 API Key: sk-ant-xxx
```

绑定消息渠道

1. Telegram

创建Bot：

1. 在Telegram搜索 `@BotFather`
2. 发送 `/newbot` 创建机器人
3. 保存Bot Token

配置：

```
openclaw channels add telegram
openclaw config set channels.telegram.botToken "your-bot-token"
openclaw gateway restart
```

2. WhatsApp

```
# 登录WhatsApp（显示二维码）
openclaw channels login whatsapp

# 用手机WhatsApp扫码
```

3. 企业微信（国内推荐）

```
# 安装企业微信插件
openclaw plugins install @m1heng-clawd/wework

# 配置
openclaw config set channels.wework '{"enabled":true,"corpId":"xxx","agentSecret":"xxx"}' --j
```

4. 飞书（国内推荐）

```
# 安装飞书插件
openclaw plugins install @m1heng-clawd/feishu

# 配置
openclaw config set channels.feishu '{"enabled":true,"appId":"cli_xxx","appSecret":"xxx"}' --
```

Windows常用命令速查

系统管理：

命令	功能
`openclaw --version`	查看版本
`openclaw status`	查看系统状态
`openclaw health`	健康检查
`openclaw update`	更新OpenClaw
`openclaw doctor`	诊断系统问题

配置管理：

命令	功能
`openclaw onboard`	初始化向导
`openclaw configure`	交互式配置
`openclaw config get <key>`	查看配置项
`openclaw config set <key> <value>`	修改配置项
`openclaw config unset <key>`	删除配置项

Linux本地部署

Linux用户：适合开发者，配置灵活。

系统要求

推荐发行版：

- Ubuntu 20.04+
- Debian 11+
- CentOS 8+

安装步骤

第一步：安装Node.js

```
# Ubuntu/Debian
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt-get install -y nodejs

# 验证安装
node --version
```

第二步：安装 OpenClaw

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

第三步：验证安装

```
openclaw --version
```

第四步：初始化配置

```
openclaw onboard
```

2.1 系统要求与准备

云端部署要求

如果选择云端部署，**无需任何本地环境**，只需要：

- 一个浏览器
- 20元/月预算
- 10分钟时间

云端一键部署

适合场景：无Mac电脑、需要24小时运行、多设备访问。

为什么选择云端部署？

云端部署相比本地部署具有多项优势，如表 2-1 所示。

表 2-1 云端部署优势

优势	说明
⚡ **秒级部署**	点几下鼠标就完成，无需配置环境
💰 **成本低**	20元/月起，比买Mac Mini便宜太多
手机可用	通过QQ、企微、飞书随时随地访问
🔒 **稳定可靠**	24小时运行，不用担心电脑关机
视频教程	官方视频手把手教学

方案对比

目前主流的云端部署方案对比如表 2-2 所示。

表 2-2 云端部署方案对比

方案	价格	带宽	推荐场景
腾讯云Lighthouse	20元/月，99元/年	20M	QQ、企微用户
火山引擎	9.9元/月，58元/年	5M	飞书用户

腾讯云Lighthouse部署（推荐）

第一步：购买服务器

1. **访问活动页面**：

```
https://cloud.tencent.com/act/pro/lighthouse-moltbot
```

2. **选择配置**：
   - 配置：2核2G
   - 带宽：20M
   - 地域：建议选择**硅谷**（国外地域访问AI模型更稳定）
   - 价格：20元/月 或 99元/年

3. **实名认证**：
   - 首次使用需要实名认证
   - 选择个人认证即可
   - 按提示完成认证

4. **完成购买**：
    - 点击"立即购买"
    - 支付20元（建议先买1个月试用）
    - 等待服务器创建完成
    - **可选**：关闭自动续费
5. **获取服务器信息**：
    - 购买完成后，点击头像 → "站内信"
    - 查看并记录：
        - 公网IP地址
        - 默认用户名（通常是 `lighthouse` ）
        - 初始密码

免费白嫖方案（可选）

如果你想免费试用3个月，可以通过 CodeBuddy 活动获取免费服务器。

步骤：

1. **注册 CodeBuddy**：
    - 国际版：https://www.codebuddy.ai/promotion/?ref=lweelxalgm（谷歌/GitHub账户）
    - 国内版：https://www.codebuddy.cn/promotion/?ref=7zucxaz7zvqi（手机号）
    - 建议使用新账号注册
2. **领取奖励**：
    - 登录后点击"实战礼" → "立刻领奖"
    - 获得1个月免费使用权
    - **累计活跃7日可再延长2个月**（每天在CodeBuddy中问候即可）
3. **重装系统为OpenClaw**：
    - 登录腾讯云控制台：https://console.cloud.tencent.com/
    - 进入"轻量应用服务器"
    - 点击"重装系统"
    - 选择"使用应用模板" → "OpenClaw"

- 选择"无需备份"，点击"确定"

4. **后续操作**：
   - 重装完成后，按照下面的步骤继续配置

第二步：连接服务器

1. **使用SSH客户端连接**：

   方式一：使用SSH客户端（推荐）

   - 下载SSH客户端（如 Xterminal、Termius、FinalShell）
   - 新建SSH连接：
     - 名称：随便取
     - 地址：公网IP地址
     - 端口：22
     - 用户名：lighthouse（或站内信中的用户名）
     - 密码：购买时设置的密码

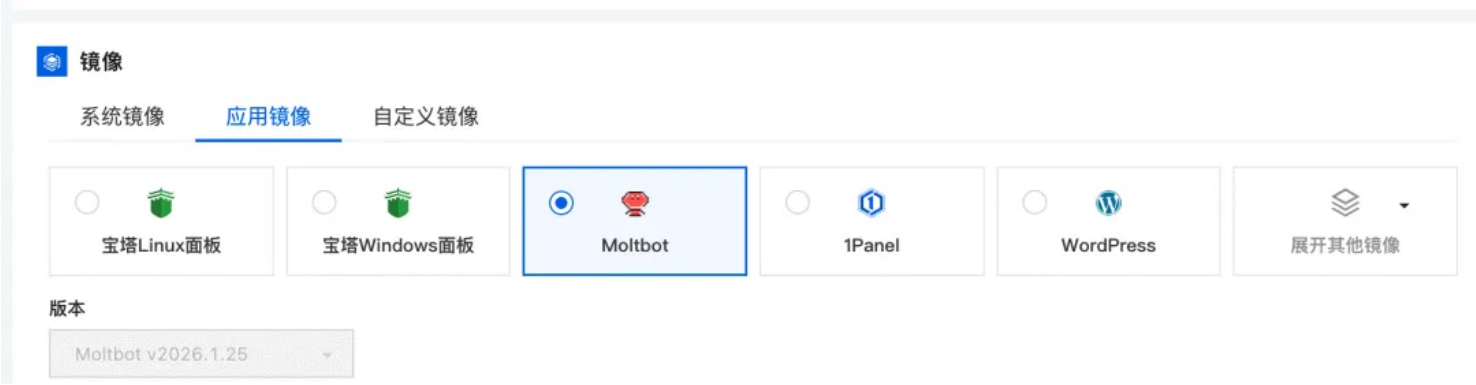
   方式二：使用网页终端

   - 在腾讯云控制台，点击实例卡片
   - 点击"登录"按钮
   - 直接在浏览器中打开终端

2. **验证OpenClaw安装**：

   ```
   openclaw --version
   ```

   如果显示版本号（如 `2026.2.9`），说明OpenClaw已预装成功。



第三步：配置大模型

1. **进入应用管理**：
   - 点击服务器卡片
   - 切换到"应用管理"标签

2. **选择模型**：
    - 推荐使用 **Kimi k2.5**（性价比最高）
    - 也可以选择其他国产大模型

```
Model/auth provider
  ○ OpenAI
  ○ Anthropic
  ○ MiniMax
  ● Moonshot AI (Kimi K2 + Kimi Coding)
  ○ Google
  ○ OpenRouter
  ○ Qwen
  ○ Z.AI (GLM 4.7)
  ○ Copilot
  ○ Vercel AI Gateway
  ○ OpenCode Zen
  ○ Xiaomi
  ○ Synthetic
  ○ Venice AI
  ○ Skip for now
```

3. **获取API Key**：

   Kimi k2.5配置（推荐）：

   ```
   1. 访问：https://platform.moonshot.cn/
   2. 注册账号并登录
   3. 进入"API管理"
   4. 点击"创建API Key"
   5. 复制API Key（格式：sk-xxx）
   ```

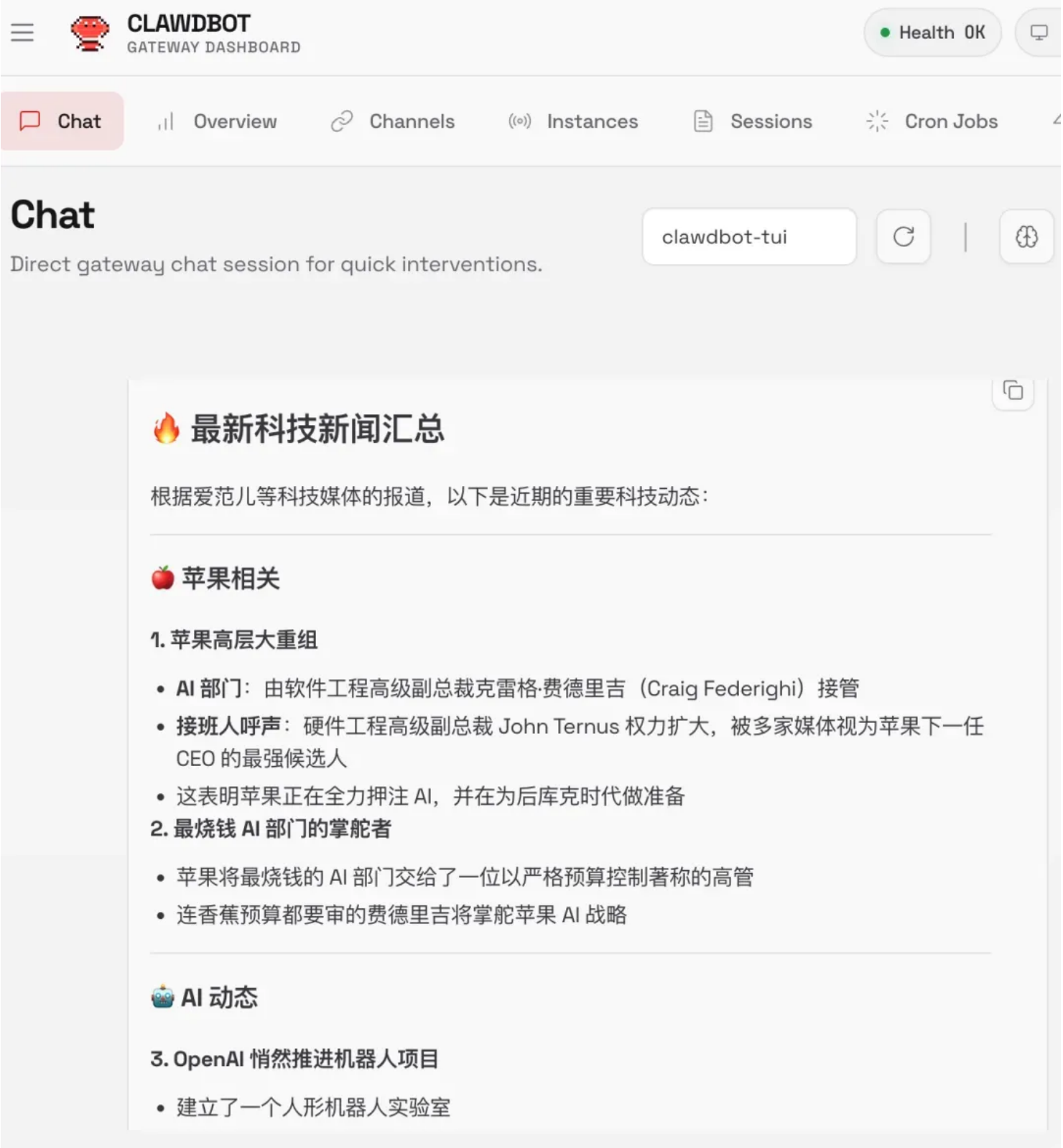
4. **填入配置**：
    - 将API Key粘贴到配置框
    - 点击"保存"
    - 等待配置生效

第四步：测试连接

1. **访问WebUI**：
    - 使用控制台提供的访问地址
    - 格式：`http://你的服务器IP:18789/?token=xxx`

2. **发送测试消息**：

   ```
   你好，能听到我说话吗？
   ```



3. **验证成功**：
    - 如果收到AI回复，说明配置成功
    - 右上角会显示使用的模型名称

火山引擎部署（更便宜）

如果你是飞书重度用户，推荐使用火山引擎：

1. **访问活动页面**：

```
https://www.volcengine.com/activity/clawdbot
```

2. **价格优势**：
    - 9.9元/月
    - 58元/年
    - 比腾讯云便宜10元
3. **配置对比**：
    - 2核2G（相同）
    - 5M带宽（腾讯云20M）
    - 适合飞书用户
4. **部署流程**：
    - 与腾讯云类似
    - 按照页面提示操作即可

百度智能云部署（0.01元试用）

> 💰 **超值试用**：百度智能云提供0.01元/月的特惠活动，适合想要低成本试用的用户。

活动信息

1. **访问活动页面**：

```
https://cloud.baidu.com/product/BCC/moltbot.html
```

2. **活动规则**：
    - 首月仅需 **0.01元**
    - 需要注册并绑定个人身份证
    - 每个账号限购一次

部署步骤

第一步：购买服务器

1. 注册百度智能云账号
2. 完成个人实名认证（绑定身份证）
3. 抢购特惠LS实例（0.01元/月）
4. 如果售罄，可以原价购买轻量应用服务器

第二步：创建实例

1. 进入轻量应用服务器LS控制台
2. 点击"创建实例"：
    - 名称：随机生成或自定义
    - 密码：**务必记住**，后续登录需要

3. 等待实例创建完成

第三步：一键配置 OpenClaw

1. **进入实例详情页**：
    - 点击实例卡片
    - 进入"应用管理"标签
2. **应用配置**：
    - 点击"一键开通"
    - 点击"一键放行"（开放防火墙端口）
    - 等待显示"已放行"
3. **模型配置**：
    - 下拉选择模型（如：文心一言、千帆大模型）
    - 点击"应用模型配置"
    - 系统会自动创建千帆API Key并配置
4. **接入方式配置**（可选）：
    - 支持接入：飞书、钉钉、企业微信、QQ
    - 选择你常用的通讯工具
    - 按照提示完成配置（详见后续章节）
5. **Skills 配置**（可选）：
    - 默认提供：百度搜索、百度百科
    - 可以按需选择并点击"应用"
    - 更多Skills可访问OpenClaw官网获取

第四步：访问WebUI

1. 点击"获取网站地址"
2. 复制访问链接
3. 在浏览器中打开，即可与OpenClaw对话

优势与限制

优势：

- 价格极低（首月0.01元）
- 一键配置，无需手动安装
- 集成百度千帆大模型
- 自动配置APIKey

限制：

- ⚠ 仅限首月优惠
- ⚠ 需要实名认证
- ⚠ 活动可能售罄

阿里云部署（可选）

阿里云也提供了OpenClaw一键部署方案：

1. **访问活动页面**：

```
https://www.aliyun.com/activity/ecs/clawdbot
```

2. **选择轻量应用服务器**：

    - 使用 OpenClaw镜像
    - 一键安装配置

3. **价格参考**：
    - 与腾讯云类似
    - 具体以活动页面为准

官方视频教程（强烈推荐）

腾讯云提供了详细的视频教程，跟着视频操作更简单：

1. **云上OpenClaw一键部署并接入企微和QQ**

    - 视频地址：https://cloud.tencent.com/developer/video/85003
    - 时长：约10分钟
    - 内容：从购买到配置完成

2. **云上OpenClaw一键部署并接入飞书和钉钉**

    - 视频地址：https://cloud.tencent.com/developer/video/85055
    - 时长：约10分钟
    - 内容：飞书和钉钉接入全流程

存量服务器部署

如果你已经有轻量服务器，可以使用AI助手对话式部署：

- **官方教程**：https://cloud.tencent.com/developer/article/2625605
- **适用场景**：已有轻量服务器
- **部署方式**：通过AI对话完成配置
- **优势**：更灵活，可自定义

云端部署常见问题

Q1: 云端部署安全吗？

- 数据存储在你的服务器上
- 只有你能访问
- 可以设置访问密码

Q2: 可以随时停止吗？

- 可以随时删除服务器
- 按使用时长计费
- 不用了就删除，不浪费钱

Q3: 手机怎么访问？

- 通过QQ、企微、飞书等平台
- 详见第9章：多平台集成

国内一键安装（推荐）

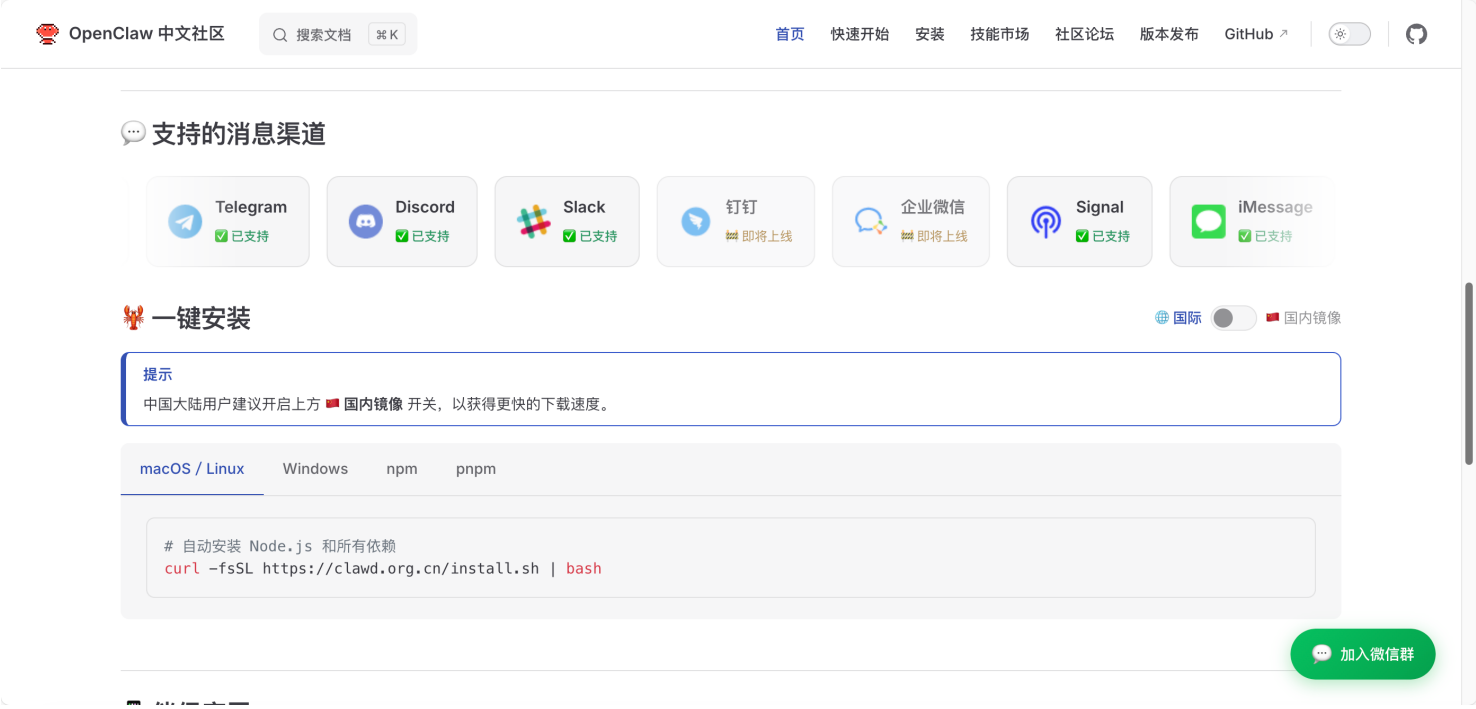
国内用户推荐：使用官方中文版一键安装脚本，速度快、配置简单。

为什么选择国内版？

国内版相比国际版具有多项本地化优势，如表 2-3 所示。

表 2-3 国内版优势

优势	说明
⚡ **速度快**	使用国内镜像源，下载速度快
中文友好	完整中文界面和提示
📦 **一键安装**	自动配置所有依赖
开箱即用	预配置国内常用服务
💰 **成本优化**	默认配置国产模型



前置要求

必需环境：

- Node.js 22.0.0+（必需）
- pnpm（可选，推荐用于源码构建）

推荐配置：

- Brave Search API 密钥（用于网络搜索）
- 可通过 `openclaw-cn configure --section web` 配置

系统要求：

- macOS：需要 Xcode / Command Line Tools（仅 CLI + Gateway 需要 Node.js）
- Windows：强烈推荐使用 WSL2（Ubuntu），原生 Windows 未经测试
- Linux：Ubuntu 20.04+、Debian、CentOS

快速开始

macOS/Linux 安装

```
# 使用国内官方安装脚本
curl -fsSL https://clawd.org.cn/install.sh | bash
```

Windows 安装

使用 PowerShell（管理员权限）：

```
# 使用国内官方安装脚本
iwr -useb https://clawd.org.cn/install.ps1 | iex
```

⚠ **Windows 用户注意**：强烈推荐使用 WSL2（Ubuntu），原生 Windows 支持有限。

WSL2 安装步骤：

```
# 1. 安装 WSL2
wsl --install

# 2. 重启电脑

# 3. 在 WSL2 中运行 Linux 安装命令
curl -fsSL https://clawd.org.cn/install.sh | bash
```

全局安装（替代方案）

如果一键脚本失败，可以使用 npm 全局安装：

```
# 使用 npm
npm install -g openclaw-cn@latest

# 或使用 pnpm（推荐）
pnpm add -g openclaw-cn@latest
```

运行入门向导

安装完成后，运行配置向导：

```
# 运行入门向导并安装后台服务
openclaw-cn onboard --install-daemon
```

配置向导流程

向导会引导你完成以下配置：

1. 选择网关模式：

- 本地网关（推荐）：Gateway 运行在本机
- 远程网关：Gateway 运行在服务器

2. 配置认证：

- OpenAI Code（Codex）订阅（OAuth）
- API 密钥（推荐用于 Anthropic）
- 也支持 `claude setup-token`

3. 选择 AI 提供商：

- 推荐：Kimi、DeepSeek、GLM-4（国产模型）
- 可选：Claude、GPT（需要 API key）

4. 配置聊天平台（可选）：

- WhatsApp：QR 扫码登录
- Telegram：输入 Bot Token
- Discord：输入 Bot Token
- Mattermost：插件令牌
- 飞书/企微/钉钉：输入应用凭证

5. 安装后台服务（推荐）：

- macOS：使用 launchd
- Linux：使用 systemd
- WSL2：使用 systemd
- 运行时：Node（推荐，WhatsApp/Telegram 必需）

6. 网关令牌：

- 向导默认生成一个令牌
- 存储在 `gateway.auth.token` 中
- 即使在回环地址上也会生成

认证配置说明

认证存储位置（重要）：

- **推荐的 Anthropic 路径**：设置 API 密钥（向导可存储）
- OAuth 凭据（旧版导入）：`~/.openclaw/credentials/oauth.json`
- 认证配置文件：`~/.openclaw/agents/<agentId>/agent/auth-profiles.json`

无头/服务器提示：

- 先在普通机器上完成 OAuth
- 然后将 `oauth.json` 复制到网关主机

启动 Gateway

如果安装了后台服务，Gateway 应该已经在运行：

```
# 检查 Gateway 状态
openclaw-cn gateway status
```

手动运行（前台）：

```
# 在前台运行，查看实时日志
openclaw-cn gateway --port 18789 --verbose
```

Dashboard 访问：

本地回环地址： `http://127.0.0.1:18789/`

如果配置了令牌，需要将其粘贴到控制界面设置中（存储为 `connect.params.auth.token` ）。

> ⚠ **Bun 警告（WhatsApp + Telegram）**：Bun 在这些渠道上有已知问题。如果使用 WhatsApp 或 Telegram，请使用 Node 运行网关。

快速验证（2分钟）

```
# 检查状态
openclaw-cn status

# 健康检查
openclaw-cn health
```

配对 + 连接聊天界面

WhatsApp（二维码登录）

```
# 登录 WhatsApp
openclaw-cn channels login
```

通过 WhatsApp → 设置 → 已连接的设备 扫描二维码。

Telegram / Discord / 其他

向导可以为你写入令牌/配置。如果手动配置：

Telegram：

```
openclaw-cn channels add \
  --channel telegram \
  --token "YOUR_BOT_TOKEN"
```

Discord：

```
openclaw-cn channels add \
  --channel discord \
  --token "YOUR_BOT_TOKEN"
```

> **Telegram 私信提示**：首次私信会返回配对码，需要批准后机器人才会响应。

私信安全（配对审批）

默认策略：未知私信会收到短代码，消息在获得批准前不会被处理。

如果首次私信没有得到回复，需要批准配对：

```
# 查看配对列表
openclaw-cn pairing list whatsapp

# 批准配对
openclaw-cn pairing approve whatsapp <code>
```

从源码运行（开发）

如果需要修改 OpenClaw 本身，可以从源码运行：

```
# 克隆仓库
git clone https://github.com/clawdbot/clawdbot.git
cd clawdbot

# 安装依赖
pnpm install

# 构建 UI（首次运行时自动安装 UI 依赖）
pnpm ui:build

# 构建项目
pnpm build

# 运行入门向导
openclaw-cn onboard --install-daemon
```

如果还没有全局安装，可以从仓库中通过 `pnpm openclaw-cn ...` 运行命令。

从源码运行 Gateway：

```
node dist/entry.js gateway --port 18789 --verbose
```

端到端验证

在新终端中，发送测试消息：

```
# 发送测试消息
openclaw-cn message send --target +15555550123 --message "Hello from OpenClaw"
```

如果 `openclaw-cn health` 显示 "no auth configured"，需要返回向导设置 OAuth/密钥认证。

调试提示：

- `openclaw-cn status --all` ：最佳的只读调试报告
- `openclaw-cn health` ：向运行中的网关请求健康快照
- `openclaw-cn status --deep` ：深度状态检查

配置文件位置

详细说明: 完整的配置文件结构和使用指南请参考 [配置文件结构完整指南]

```
# 主配置文件
~/.openclaw/openclaw.json

# 认证配置
~/.openclaw/agents/<agentId>/agent/auth-profiles.json

# OAuth 凭据（旧版）
~/.openclaw/credentials/oauth.json

# 日志文件
~/.openclaw/logs/gateway.log
```

国内版特色功能

1. 预配置国产模型：

- Kimi（月之暗面）
- DeepSeek（深度求索）
- GLM-4（智谱 AI）
- 通义千问（阿里）
- 文心一言（百度）

2. 国内平台集成：

- 飞书（字节跳动）
- 企业微信（腾讯）
- 钉钉（阿里）
- QQ（腾讯）

3. 优化的网络配置：

- 使用国内镜像源
- 优化 API 访问速度
- 支持代理配置

常见问题

Q1: 安装失败怎么办？

```
# 检查 Node.js 版本（需要 22+）
node --version

# 如果版本过低，使用 nvm 升级
nvm install 22
nvm use 22
```

Q2: 如何更新到最新版本？

```
# 重新运行安装脚本
curl -fsSL https://clawd.org.cn/install.sh | bash
```

Q3: 如何卸载？

```
# 停止服务
openclaw-cn gateway stop

# 卸载
npm uninstall -g openclaw-cn

# 删除配置（可选）
rm -rf ~/.openclaw
```

Q4: 支持哪些系统？

- macOS 12+
- Linux（Ubuntu 20.04+、Debian、CentOS）
- Windows 10/11（通过 WSL2）

Q5: 配置向导卡住怎么办？

```
# 按 Ctrl+C 中断

# 检查网关是否运行
openclaw-cn gateway status

# 重新启动网关并重试
openclaw-cn gateway restart
openclaw-cn onboard
```

Q6: 健康检查显示 "no auth configured"

需要配置认证：

```
# 重新运行向导
openclaw-cn onboard

# 或手动配置 API 密钥
openclaw-cn configure --section auth
```

下一步（可选，但很棒）

- macOS 菜单栏应用 + 语音唤醒：macOS 应用
- iOS/Android 节点（Canvas/摄像头/语音）：节点
- 远程访问（SSH 隧道 / Tailscale 服务）：远程访问

- 始终在线 / VPN 设置：Tailscale

下一步

安装完成后，可以：

1. 配置 AI 模型（见下文"API配置指南"）
2. 连接聊天平台（见第9章：多平台集成）
3. 安装 Skills（见第8章：Skills扩展）
4. 开始使用（见第3章：快速上手)

Cloudflare Workers 部署（进阶）

> ☁️ **全球 CDN 加速**：使用 Cloudflare Workers 部署 OpenClaw，享受全球边缘网络加速。

为什么选择 Cloudflare Workers？

Cloudflare Workers 提供全球边缘网络部署能力，如表 2-4 所示。

表 2-4 Cloudflare Workers 优势

优势	说明
🌍 **全球加速**	部署在 Cloudflare 全球边缘网络
💰 **成本可控**	5美元/月起步，24小时在线
🔒 **安全可靠**	内置 Zero Trust 安全认证
⚡ **快速部署**	一键部署，10分钟完成
📦 **无需服务器**	Serverless 架构，无需维护

前置要求

必需条件：

- Cloudflare 账号
- Workers Paid 计划（5美元/月）
- 信用卡（用于订阅付费计划）

成本说明：

- 基础费用：5美元/月（起步价）
- 高频使用可能产生额外费用
- 作为 24 小时在线的 AI 服务，月成本在可接受范围内

> **成本参考**：详见 GitHub 讨论：What's the cost running it 24/7 for a month

部署流程

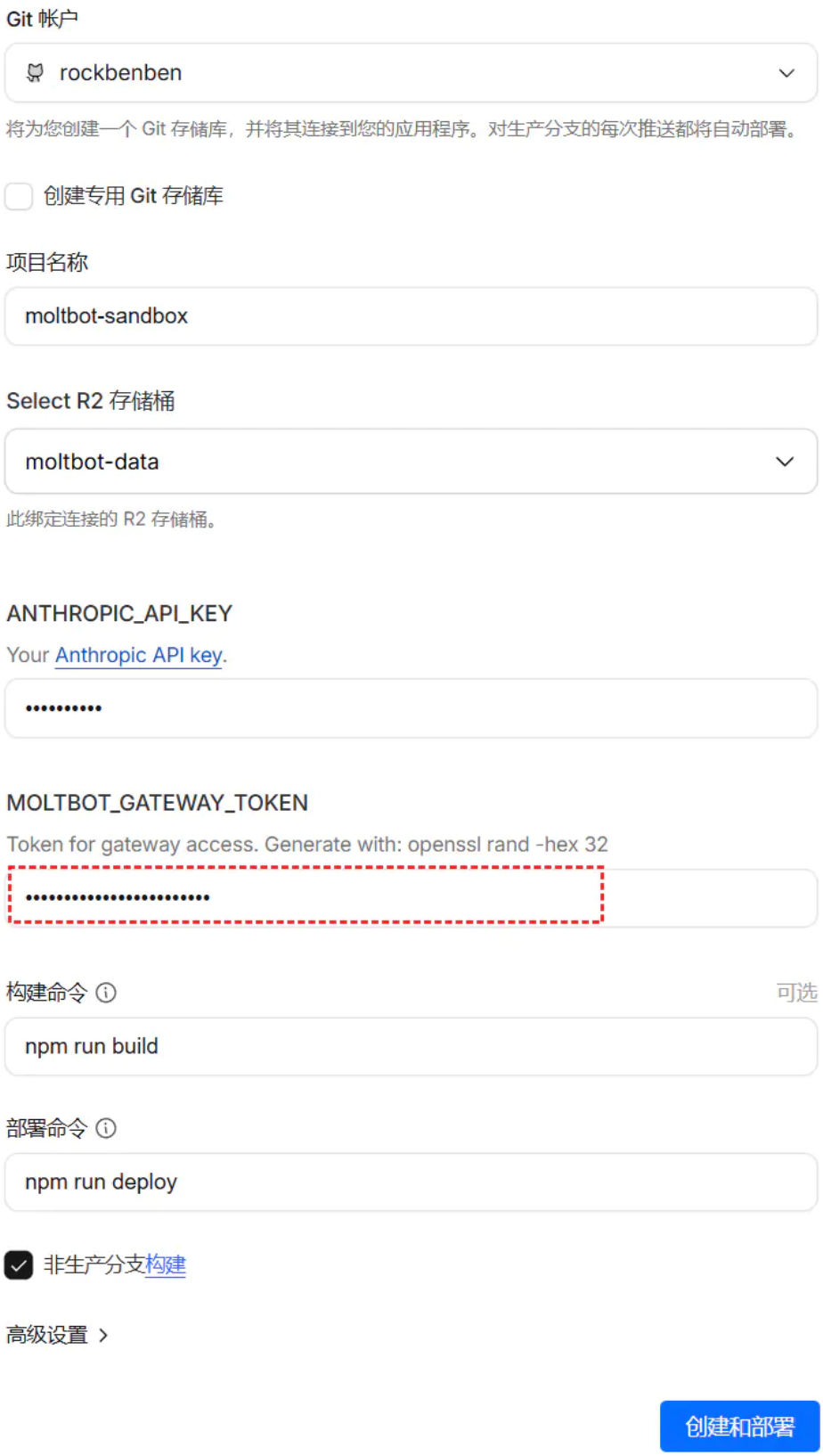
第一步：一键部署 Moltworker

1. **点击部署按钮**：

```
https://deploy.workers.cloudflare.com/?url=https://github.com/cloudflare/moltworker
```

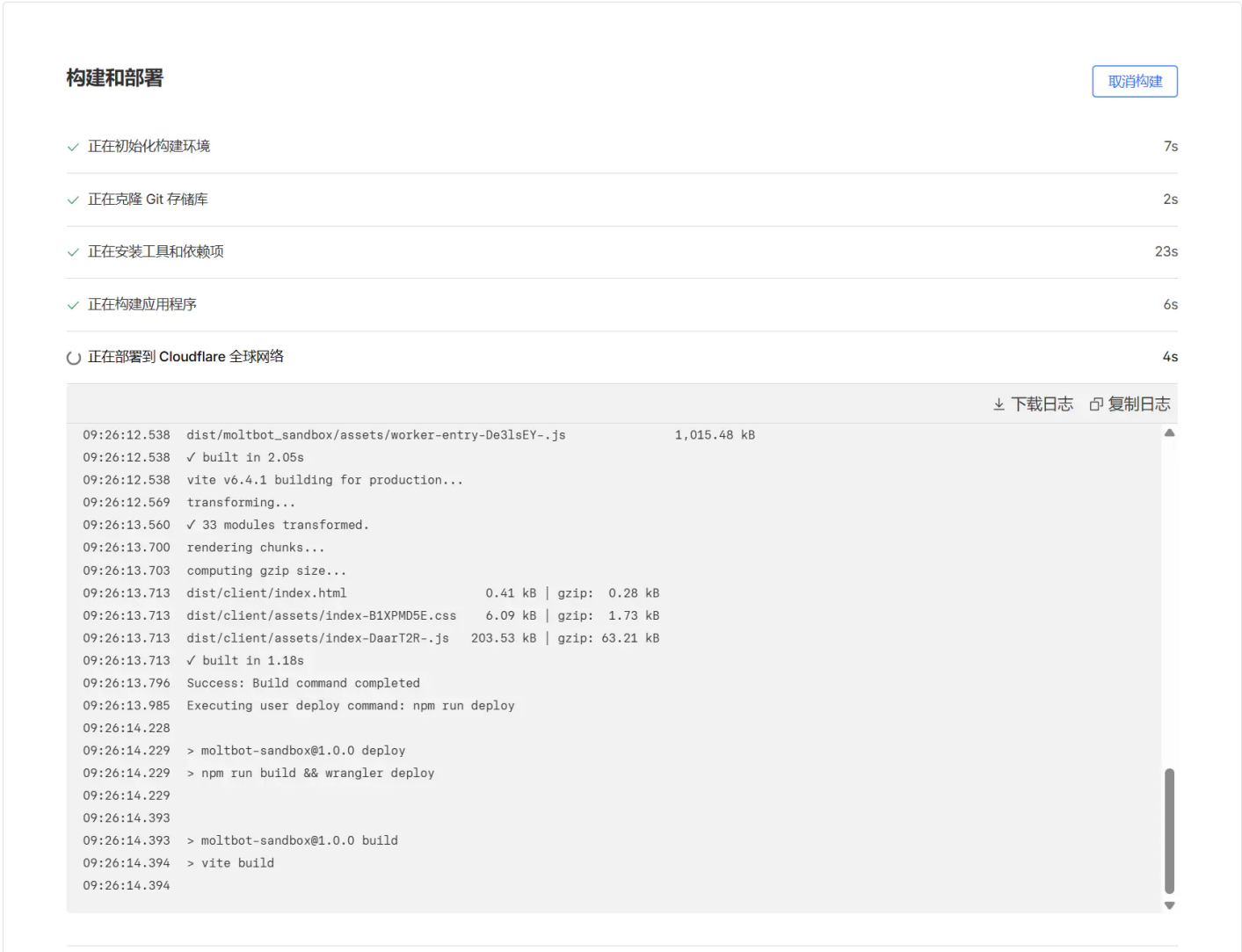
2. **配置 Gateway Token**：

   - 务必修改并妥善保存 `MOLTBOT_GATEWAY_TOKEN`
   - 这是后续进入管理后台的唯一凭证
   - 建议使用强密码生成器



第二步：等待构建

- 部署过程约需 10 分钟
- 可点击「继续处理项目」跳过等待页面
- 构建完成后会自动跳转到项目页面



第三步：配置 Access（Zero Trust）

访问网页界面需要配置 `CF_ACCESS_AUD` 和 `CF_ACCESS_TEAM_DOMAIN` 两个变量。

1. 创建应用：

- 进入 Zero Trust → Access → Applications
- 添加一个 Self-hosted 应用

2. 设置域名：

- 子域默认为 `moltbot-sandbox`
- 域名可使用 Cloudflare 分配的 Worker 域名或自定义域名
- Session Duration（会话时间）建议设置长一些，避免频繁登录

3. 配置策略：

- 系统会自动创建 `moltbot-sandbox - Production` 策略
- 默认通过邮箱验证码登录

4. 获取配置变量：

CF_ACCESS_AUD：

- 保存应用后，点击右侧「⋮」编辑
- 在应用程序受众（AUD）标签页找到 Application Audience (AUD)

CF_ACCESS_TEAM_DOMAIN：

- 进入 Zero Trust → Settings
- 团队域名格式：`xxxxxx.cloudflareaccess.com`

第四步：配置 R2 对象存储

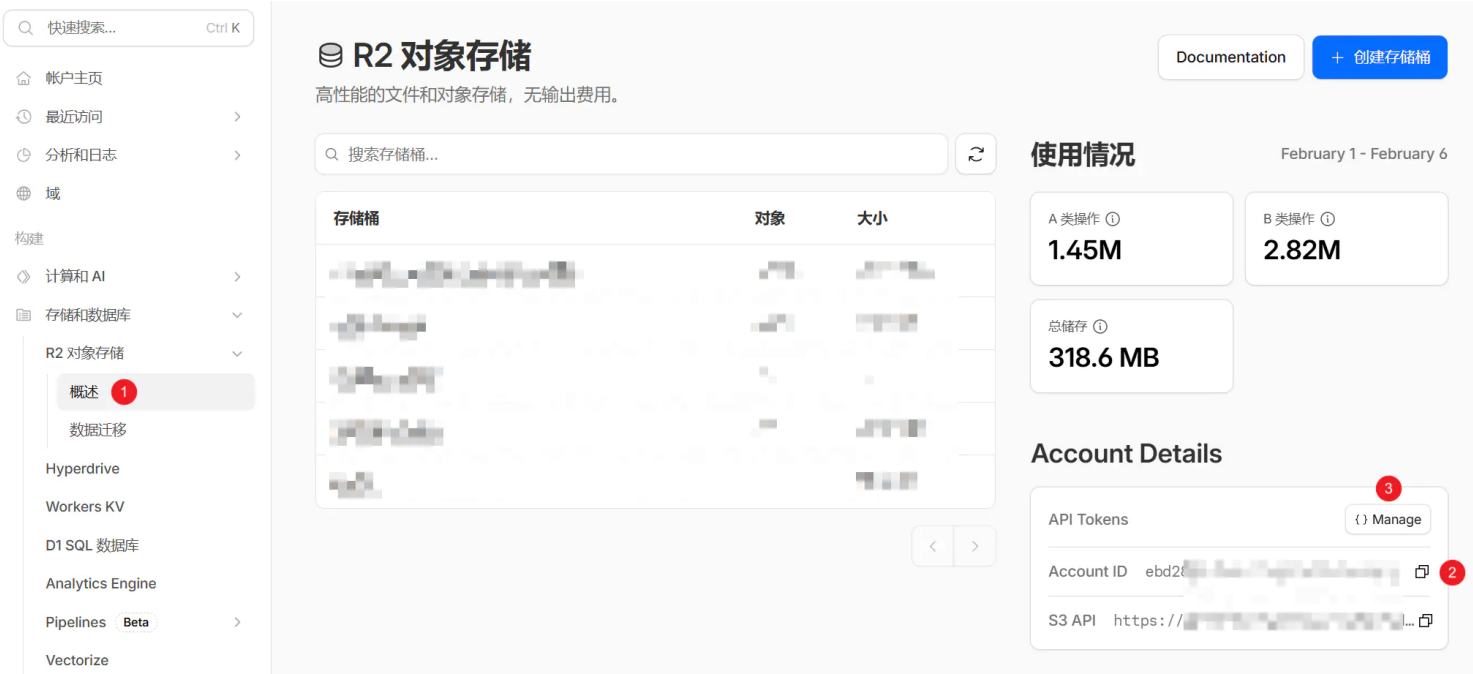
OpenClaw 需要 R2 来存储状态，需配置以下三个变量：

- `CF_ACCOUNT_ID`
- `R2_ACCESS_KEY_ID`
- `R2_SECRET_ACCESS_KEY`

操作步骤：

1. 获取 Account ID：

- 在 Cloudflare 侧边栏进入 R2 → Overview
- 右侧 Account Details 中的 Account ID 即为 `CF_ACCOUNT_ID`

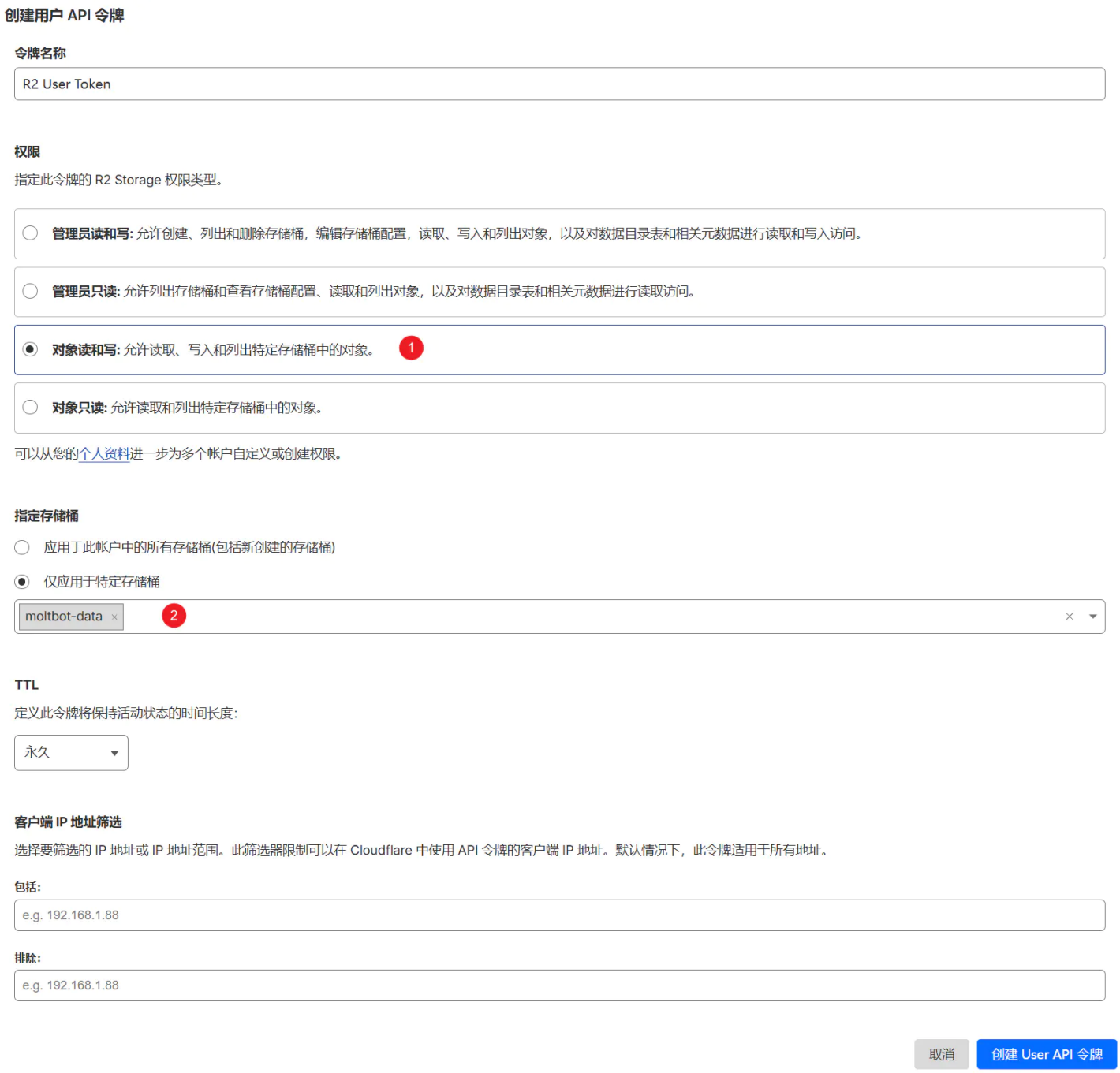


2. 创建 API 令牌：

- 点击 Manage R2 API Tokens
- 选择 Create API Token

3. 设置权限：

- 权限选择 Object Read & Write
- 建议范围通过 Specific Bucket 限制在 `moltbot-data`



4. 保存密钥：

- 创建成功后，记录 Access Key ID 和 Secret Access Key

为 S3 客户端使用以下凭据：

访问密钥 ID

```
29e9c3180cbee0720519f9f06b98ed1d
```

单击以复制

机密访问密钥

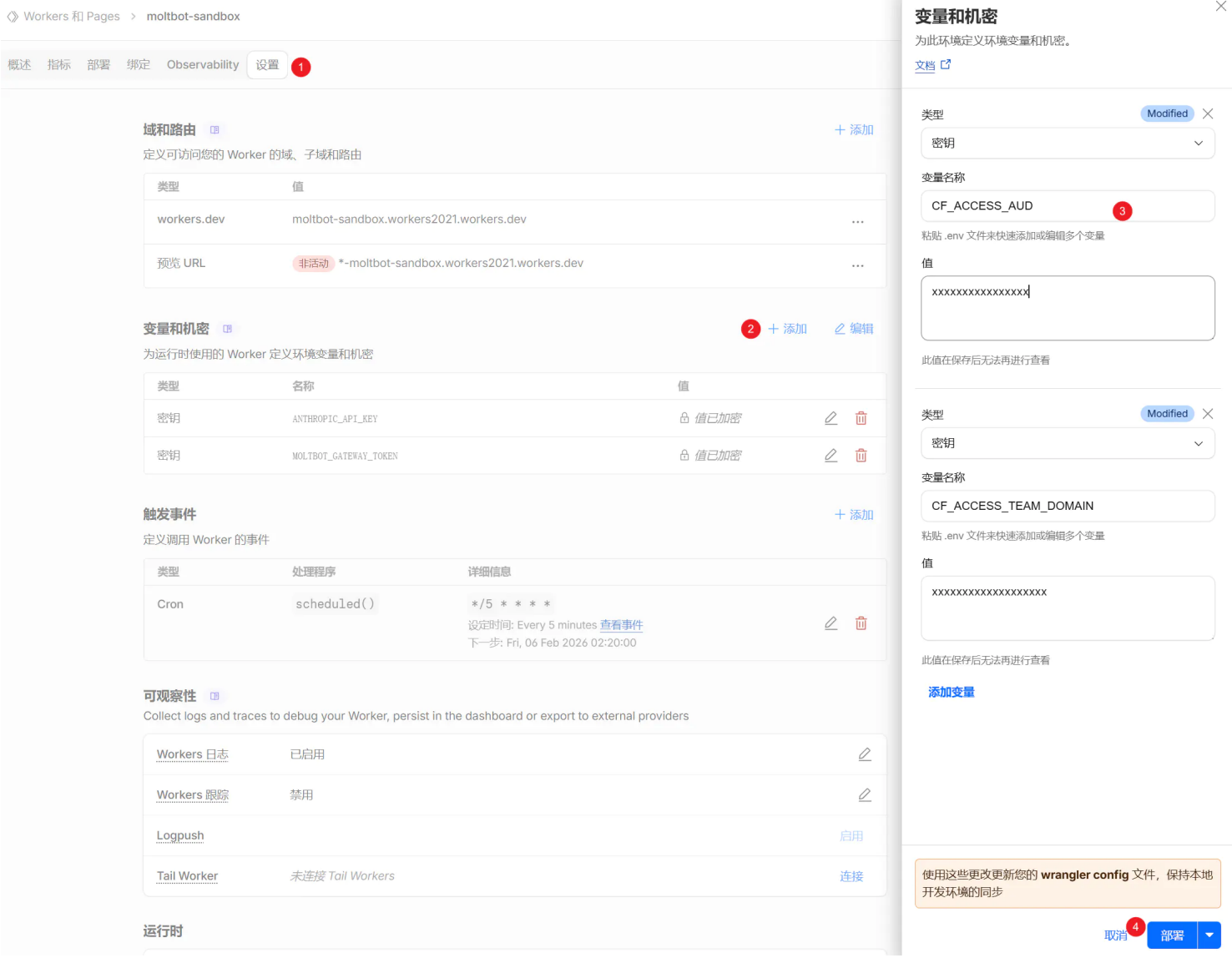
```
e60efdd045061c41ad5d217b767c8f5a2a41a427e5403579eda558924bda8ee5
```

单击以复制

> ⚠ **重要提示**：修改 Token 时请务必核对变量名称。如果不慎修改了 Build Token，会导致 Worker 构建失败。

第五步：注入变量并重启

1. **进入设置**：
   - Workers → Settings → Variables and Secrets
2. **填入变量**：
   - `MOLTBOT_GATEWAY_TOKEN`（第一步设置的）
   - `CF_ACCESS_AUD`（第三步获取的）
   - `CF_ACCESS_TEAM_DOMAIN`（第三步获取的）
   - `CF_ACCOUNT_ID`（第四步获取的）
   - `R2_ACCESS_KEY_ID`（第四步获取的）
   - `R2_SECRET_ACCESS_KEY`（第四步获取的）
3. **重新部署**：
   - 点击 Deploy 重新部署
   - 等待部署完成



访问与管理

部署完成后，可通过以下地址访问：

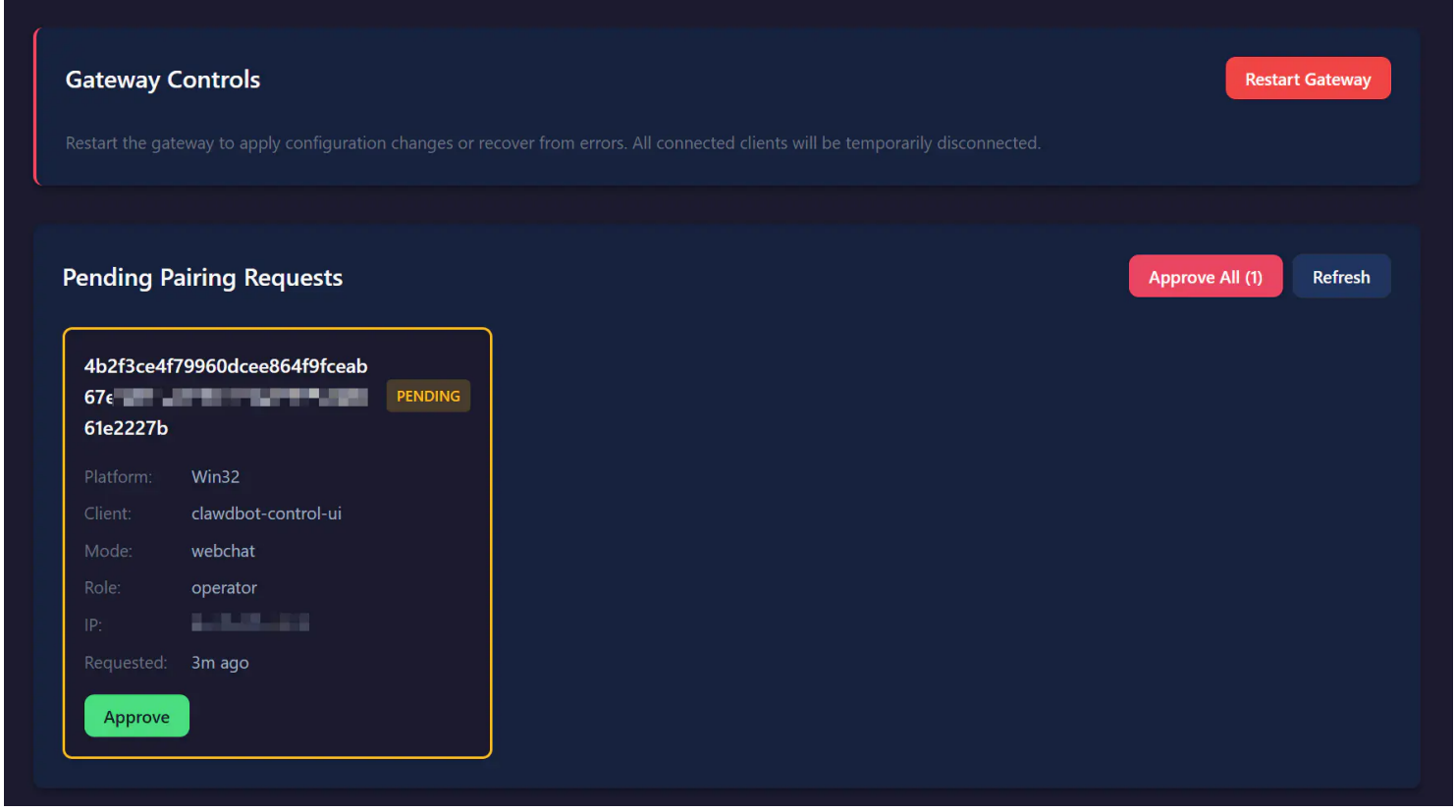
访问 Worker（需要 token）：

```
https://moltbot-sandbox.xxxxxxxx.workers.dev?token=MOLTBOT_GATEWAY_TOKEN
```

管理后台（需要邮箱验证）：

```
https://moltbot-sandbox.xxxxxxxx.workers.dev/_admin/
```

通过 Cloudflare Access 的邮箱验证码验证后，即可进入管理后台并接受 Pairing Requests。



基础使用

查看或切换模型

```
# 查看当前模型
/model

# 切换模型
/model minimax/MiniMax-M2.1
```

设置开机自启命令

为了避免 Worker 重启后模型被重置，建议设置开机自启命令：

```
set model minimax/MiniMax-M2.1
```

远程终端连接

```
# 登录到 Gateway
openclaw gateway login --url https://moltbot-sandbox.xxxxxxxx.workers.dev

# 配置 Skills
openclaw configure --section skills
```

避坑指南

问题 1：模型配置报错

症状：通过配置文件修改默认模型后报错

原因：国内 AI 服务商通常区分国内与海外端点，Cloudflare Workers 环境下配置文件修改容易出错

解决方案：

- 直接通过开机命令强制指定模型
- 不要依赖配置文件或后台 UI
- 使用 `set model` 命令设置开机自启

问题 2：Worker 构建失败

症状：部署后 Worker 无法启动

原因：不慎修改了 Build Token

解决方案：

- 检查所有变量名称是否正确
- 确保没有修改 Build Token
- 重新部署

问题 3：无法访问管理后台

症状：访问 `/_admin/` 时无法登录

原因：Zero Trust 配置不正确

解决方案：

- 检查 `CF_ACCESS_AUD` 和 `CF_ACCESS_TEAM_DOMAIN` 是否正确
- 确认邮箱验证码是否正确
- 检查 Session Duration 设置

成本估算

项目	费用	说明
Workers Paid 计划	5美元/月	基础费用
额外请求费用	按量计费	高频使用时产生
R2 存储	免费额度内	通常不会超出
总计	5-10美元/月	取决于使用频率

适用场景

推荐使用：

- 想低成本尝试 OpenClaw
- 已有 Cloudflare 付费订阅
- 需要全球 CDN 加速

- 不想维护服务器

不推荐使用：

- 期望开箱即用
- 没有技术背景
- 需要复杂的自动化流程
- 预算非常有限

总结

Cloudflare Workers + OpenClaw 是一个低成本的尝鲜方案，适合：

- 未体验过 Agent 自动化，想低成本试手
- 已有 Cloudflare 付费订阅，资源闲置
- 需要全球 CDN 加速的场景

但需要注意：

- OpenClaw 目前还不是一个能「即刻提升效率」的工具
- 更像是一个为 AI 自动化搭建的系统底座
- 如果没有明确的、可标准化的长流程需求，可能只会带来维护成本

下一步：

- 配置 AI 模型（见下文"API配置指南"）
- 配置通讯渠道（见第9章：多平台集成）
- 安装 Skills（见第8章：Skills扩展）

Docker 部署（可选）

> **开发者选项**：Docker 部署适合需要环境隔离的场景。

为什么选择 Docker？

Docker 部署提供环境隔离和便捷管理，如表 2-5 所示。

表 2-5 Docker 部署优势

优势	说明
🔒 **环境隔离**	不影响系统环境，干净整洁
📦 **一键部署**	无需配置依赖，开箱即用
易于更新	一条命令完成更新
跨平台	Windows/macOS/Linux 统一方案
快速启动	5分钟完成部署

前置要求

安装 Docker：

macOS：

```
# 下载 Docker Desktop
# 访问：https://www.docker.com/products/docker-desktop

# 或使用 Homebrew
brew install --cask docker
```

Windows：

```
# 下载 Docker Desktop
# 访问：https://www.docker.com/products/docker-desktop

# 安装 WSL2（如果还没安装）
wsl --install
```

Linux (Ubuntu)：

```
# 安装 Docker
curl -fsSL https://get.docker.com | sh

# 启动 Docker 服务
sudo systemctl start docker
sudo systemctl enable docker

# 添加当前用户到 docker 组
sudo usermod -aG docker $USER
```

验证安装：

```
docker --version
# 应显示：Docker version 24.x.x
```

快速开始

方式一：一键脚本部署（推荐新手）

最简单的方式，一条命令搞定所有配置！

```
curl -fsSL https://clawd.org.cn/install.sh | bash
```

这个脚本会自动：

- 检查 Docker 环境
- 下载镜像（使用国内镜像：`jiulingyun803/openclaw-cn:latest`）
- 配置环境变量
- 启动容器
- 运行配置向导
- 生成网关令牌

完成后，在浏览器打开 `http://127.0.0.1:18789/` 即可使用。

脚本后续操作：

- 按照提示输入渠道信息（可选）
- 将生成的令牌复制到 Web UI 登录

方式二：手动 Docker Compose 部署（适合进阶用户）

如果一键脚本不适用，或需要自定义配置，按以下步骤操作。

步骤 1：创建工作目录

```
mkdir -p ~/openclaw-docker
cd ~/openclaw-docker
```

步骤 2：创建 `.env` 环境文件

```
cat > .env << 'EOF'
# 镜像配置（使用国内镜像）
OPENCLAW_IMAGE=jiulingyun803/openclaw-cn:latest

# 数据目录
OPENCLAW_CONFIG_DIR=./data/.openclaw
OPENCLAW_WORKSPACE_DIR=./data/clawd

# 网关配置
OPENCLAW_GATEWAY_PORT=18789
OPENCLAW_BRIDGE_PORT=18790
OPENCLAW_GATEWAY_BIND=lan
OPENCLAW_GATEWAY_TOKEN=your-secure-token-here

# Claude 集成（可选）
CLAUDE_AI_SESSION_KEY=
CLAUDE_WEB_SESSION_KEY=
CLAUDE_WEB_COOKIE=
EOF
```

步骤 3：创建 `docker-compose.yml` 文件

```
services:
  openclaw-cn-gateway:
```

```
    image: ${OPENCLAW_IMAGE:-jiulingyun803/openclaw-cn:latest}
    user: node:node
    environment:
      HOME: /home/node
      TERM: xterm-256color
      OPENCLAW_GATEWAY_TOKEN: ${OPENCLAW_GATEWAY_TOKEN}
      CLAUDE_AI_SESSION_KEY: ${CLAUDE_AI_SESSION_KEY}
      CLAUDE_WEB_SESSION_KEY: ${CLAUDE_WEB_SESSION_KEY}
      CLAUDE_WEB_COOKIE: ${CLAUDE_WEB_COOKIE}
    volumes:
      - ${OPENCLAW_CONFIG_DIR:-./data/.openclaw}:/home/node/.openclaw
      - ${OPENCLAW_WORKSPACE_DIR:-./data/clawd}:/home/node/clawd
    ports:
      - "${OPENCLAW_GATEWAY_PORT:-18789}:18789"
      - "${OPENCLAW_BRIDGE_PORT:-18790}:18790"
    init: true
    restart: unless-stopped
    command:
      [
        "node",
        "dist/index.js",
        "gateway",
        "--bind",
        "${OPENCLAW_GATEWAY_BIND:-lan}",
        "--port",
        "${OPENCLAW_GATEWAY_PORT:-18789}"
      ]

  openclaw-cn-cli:
    image: ${OPENCLAW_IMAGE:-jiulingyun803/openclaw-cn:latest}
    user: node:node
    environment:
      HOME: /home/node
      TERM: xterm-256color
      BROWSER: echo
      CLAUDE_AI_SESSION_KEY: ${CLAUDE_AI_SESSION_KEY}
      CLAUDE_WEB_SESSION_KEY: ${CLAUDE_WEB_SESSION_KEY}
      CLAUDE_WEB_COOKIE: ${CLAUDE_WEB_COOKIE}
    volumes:
      - ${OPENCLAW_CONFIG_DIR:-./data/.openclaw}:/home/node/.openclaw
      - ${OPENCLAW_WORKSPACE_DIR:-./data/clawd}:/home/node/clawd
    stdin_open: true
    tty: true
    init: true
    entrypoint: ["node", "dist/index.js"]
```

步骤 4：启动容器

```
# 拉取最新镜像
docker compose pull

# 启动网关（后台运行）
docker compose up -d openclaw-cn-gateway
```

```
# 查看日志（可选）
docker compose logs -f openclaw-cn-gateway
```

步骤 5：运行配置向导

```
docker compose run --rm openclaw-cn-cli onboard
```

配置向导会提示你：

- 选择网关后端（Claude、Gemini 等）
- 配置 Feishu、Telegram 等渠道
- 生成和保存配置

步骤 6：访问 Web UI

打开浏览器访问：`http://127.0.0.1:18789/`

将配置向导生成的令牌复制到登录页面即可。

环境变量详解

变量	含义	默认值	必需	说明
OPENCLAW_IMAGE	Docker 镜像名称	jiulingyun803/openclaw-cn:latest		使用国内镜像，也可指定版本号
OPENCLAW_CONFIG_DIR	配置文件目录	./data/.openclaw		OpenClaw 配置和凭证存储位置
OPENCLAW_WORKSPACE_DIR	工作空间目录	./data/clawd		代理工作文件存储位置
OPENCLAW_GATEWAY_PORT	网关端口号	18789		访问 Web UI 的端口
OPENCLAW_BRIDGE_PORT	桥接端口号	18790		用于客户端连接的端口
OPENCLAW_GATEWAY_BIND	网关绑定地址	lan		localhost（仅本机）/ lan（局域网）/ 0.0.0.0（公网，⚠ 谨慎使用）
OPENCLAW_GATEWAY_TOKEN	网关认证令牌	自动生成		Web UI 登录令牌（可自定义或留空自动生成）
CLAUDE_AI_SESSION_KEY	Claude.ai 会话密钥	空		⚠ 仅使用 Claude AI 作为后端时填写
CLAUDE_WEB_SESSION_KEY	Claude Web 会话密钥	空		⚠ 仅使用 Claude Web 版时填写
CLAUDE_WEB_COOKIE	Claude Web Cookie	空		⚠ 仅使用 Claude Web 版时填写

环境变量设置方式：

方式 A：编辑 `.env` 文件（推荐）

```
# 编辑 .env 文件
nano .env

# docker compose 会自动读取
docker compose up -d
```

方式 B：命令行设置

```
export OPENCLAW_GATEWAY_PORT=18789
docker compose up -d
```

方式 C：命令行临时覆盖

```
docker compose -e OPENCLAW_GATEWAY_PORT=8080 up -d
```

常用操作

查看网关状态

```
# 检查容器是否运行
docker compose ps

# 查看网关日志
docker compose logs openclaw-cn-gateway

# 实时查看日志（持续跟踪）
docker compose logs -f openclaw-cn-gateway
```

配置渠道

通过 CLI 容器配置各类渠道：

Telegram（需要机器人令牌）：

```
docker compose run --rm openclaw-cn-cli channels add \
  --channel telegram \
  --token "YOUR_BOT_TOKEN"
```

Discord（需要机器人令牌）：

```
docker compose run --rm openclaw-cn-cli channels add \
  --channel discord \
  --token "YOUR_BOT_TOKEN"
```

WhatsApp（QR 扫码）：

```
docker compose run --rm openclaw-cn-cli channels login
```

Feishu（需要 App ID 和 Secret）：

```
docker compose run --rm openclaw-cn-cli onboard
# 按提示输入信息
```

重新配置

```
# 重新运行配置向导
docker compose run --rm openclaw-cn-cli onboard

# 查看当前配置
docker compose run --rm openclaw-cn-cli config get
```

重启网关

```
# 重启网关容器
docker compose restart openclaw-cn-gateway

# 停止网关
docker compose down

# 重新启动
docker compose up -d openclaw-cn-gateway
```

更新到最新版本

```
# 拉取最新镜像
docker compose pull

# 重启容器（自动使用新镜像）
docker compose up -d openclaw-cn-gateway
```

清理数据（谨慎操作）

```
# 停止并删除容器
docker compose down

# 删除本地数据目录
rm -rf ./data/

# 删除本地镜像（可选）
docker rmi jiulingyun803/openclaw-cn:latest
```

数据持久化

Docker 容器的数据存储在工作目录的 `data` 文件夹：

```
~/openclaw-docker/data/
├── .openclaw/          # 配置文件
│   ├── openclaw.json   # 主配置
│   └── logs/           # 日志文件
└── clawd/              # 工作空间
    └── workspace/      # 代理工作文件
```

备份数据：

```
# 备份配置和数据
tar -czf openclaw-backup-$(date +%Y%m%d).tar.gz ./data

# 恢复数据
tar -xzf openclaw-backup-20260210.tar.gz
```

Docker 部署常见问题

问题 1：容器无法启动

症状： `docker compose up` 后容器立即退出

解决方案：

```
# 查看详细错误日志
docker compose logs openclaw-cn-gateway

# 检查端口是否被占用
sudo netstat -ltnp | grep 18789
# macOS 使用：lsof -i :18789

# 如果被占用，修改 OPENCLAW_GATEWAY_PORT
# 编辑 .env，将端口改为其他（如 18790）
nano .env
```

问题 2：权限拒绝（Permission Denied）

症状： `Error: EACCES: permission denied, mkdir ...`

解决方案：

```
# 确保数据目录存在且权限正确
mkdir -p ./data/.openclaw ./data/clawd
chmod 755 ./data/.openclaw ./data/clawd

# 如果使用了宿主机路径，确保目录可写
chmod 777 ./data
```

问题 3：无法访问 Web UI

症状：浏览器访问 `http://127.0.0.1:18789` 无响应

解决方案：

```
# 检查容器是否运行
docker compose ps

# 检查网关日志
docker compose logs openclaw-cn-gateway

# 验证端口是否正确
# 如果 OPENCLAW_GATEWAY_PORT=18789，则访问 :18789
# 如果改了端口，访问对应的新端口

# 检查防火墙设置
# macOS
sudo pfctl -d  # 临时关闭防火墙测试

# Linux
sudo ufw status
sudo ufw allow 18789
```

问题 4：配置向导卡住

症状：`docker compose run --rm openclaw-cn-cli onboard` 无反应

解决方案：

```
# 按 Ctrl+C 中断

# 检查网关是否运行
docker compose logs openclaw-cn-gateway

# 重新启动网关并重试
docker compose restart openclaw-cn-gateway
docker compose run --rm openclaw-cn-cli onboard
```

问题 5：镜像拉取失败（403 错误）

症状：`docker pull openclaw/openclaw:latest` 返回 403 错误

解决方案：

```
# 使用国内镜像（推荐）
docker pull jiulingyun803/openclaw-cn:latest

# 或在 .env 文件中指定国内镜像
echo "OPENCLAW_IMAGE=jiulingyun803/openclaw-cn:latest" >> .env
```

```
# 重新拉取
docker compose pull
```

问题 6：网络超时

症状：拉取镜像或访问 API 时网络超时

解决方案：

```
# 配置 Docker 镜像加速（国内用户）
# 编辑 Docker 配置
sudo nano /etc/docker/daemon.json

# 添加镜像加速器
{
  "registry-mirrors": [
    "https://docker.mirrors.ustc.edu.cn",
    "https://hub-mirror.c.163.com"
  ]
}

# 重启 Docker
sudo systemctl restart docker

# macOS 用户在 Docker Desktop 设置中添加镜像加速器
```

问题 7：数据丢失

症状：重启容器后配置和数据丢失

解决方案：

```
# 确保使用了数据卷挂载
# 检查 docker-compose.yml 中的 volumes 配置

# 查看数据是否存在
ls -la ./data/.openclaw
ls -la ./data/clawd

# 如果数据丢失，从备份恢复
tar -xzf openclaw-backup-20260210.tar.gz
```

问题 8：性能问题

症状：容器运行缓慢或占用资源过高

解决方案：

```
# 限制资源使用（编辑 docker-compose.yml）
services:
  openclaw-cn-gateway:
    # ... 其他配置
    deploy:
      resources:
        limits:
          cpus: '2'
          memory: 2G
        reservations:
          cpus: '1'
          memory: 1G

# 重启容器
docker compose up -d openclaw-cn-gateway
```

从一键脚本迁移到手动配置

如果想从一键脚本切换到手动配置（或反之）：

```
# 停止现有容器
docker compose down

# 备份现有配置
cp -r ~/.openclaw ~/.openclaw.backup

# 更新 .env 和 docker-compose.yml

# 重新启动
docker compose up -d openclaw-cn-gateway
```

配置会自动保留在数据目录中，无需重新设置。

Docker 部署优势总结

环境隔离：不影响系统环境
快速部署：5分钟完成
易于管理：一条命令更新
跨平台：统一部署方案
可扩展：支持多实例部署
国内优化：使用国内镜像，下载速度快

推荐使用场景：

- 开发者本地测试
- 服务器部署
- 多环境隔离
- 快速体验 OpenClaw

下一步：

- 配置 API 模型（见下文"API配置指南"）
- 配置通讯渠道（见第9章：多平台集成）
- 安装 Skills（见第8章：Skills扩展）

更新和维护

> **保持最新**：定期更新 OpenClaw 以获得新功能和安全修复。

检查更新

```
# 检查当前版本
openclaw --version

# 检查最新版本
curl -s https://api.github.com/repos/openclaw/openclaw/releases/latest | grep tag_name
```

本地安装更新

```
# 方式一：使用安装脚本
curl -fsSL https://openclaw.ai/install.sh | bash

# 方式二：手动更新
cd ~/openclaw
git pull origin main
pnpm install
pnpm build
```

Docker 更新

```
# 拉取最新镜像
docker pull openclaw/openclaw:latest

# 停止并删除旧容器
docker stop openclaw
docker rm openclaw

# 启动新容器
docker run -d \
  --name openclaw \
  -p 18789:18789 \
  -v ~/.openclaw:/root/.openclaw \
  --restart unless-stopped \
  openclaw/openclaw:latest
```

备份数据

本地安装备份：

```
# 备份配置和数据
tar -czf openclaw-backup-$(date +%Y%m%d).tar.gz ~/.openclaw

# 恢复数据
tar -xzf openclaw-backup-20260210.tar.gz -C ~/
```

Docker 备份：

```
# 备份数据卷
docker run --rm \
  -v ~/.openclaw:/data \
  -v $(pwd):/backup \
  alpine tar czf /backup/openclaw-backup-$(date +%Y%m%d).tar.gz /data

# 恢复数据
docker run --rm \
  -v ~/.openclaw:/data \
  -v $(pwd):/backup \
  alpine tar xzf /backup/openclaw-backup-20260210.tar.gz -C /
```

监控和日志

查看日志：

```
# 本地安装
tail -f ~/.openclaw/logs/gateway.log

# Docker
docker logs -f openclaw
```

监控指标：

```
# 查看系统状态
openclaw gateway status

# 查看资源使用
openclaw stats

# 查看 API 消耗
openclaw stats api
```

故障排查

常见问题：

1. **Gateway 无法启动**

```
# 查看日志
openclaw logs

# 检查端口占用
lsof -i :18789

# 重启 Gateway
openclaw gateway restart
```

2. **API 连接失败**

```
# 测试 API 连接
openclaw test api

# 检查 API Key
openclaw config get models.providers
```

3. **性能问题**

```
# 清理缓存
openclaw cache clear

# 重启服务
openclaw gateway restart
```

卸载

本地安装卸载：

```
# 停止服务
openclaw gateway stop

# 删除文件
rm -rf ~/.openclaw
rm -rf ~/openclaw

# 删除命令
npm uninstall -g openclaw
```

Docker 卸载：

```
# 停止并删除容器
docker stop openclaw
docker rm openclaw

# 删除镜像
docker rmi openclaw/openclaw
```

```
# 删除数据
rm -rf ~/.openclaw
```

API配置指南

OpenClaw需要连接AI模型才能工作，推荐使用**国产大模型**，性价比高。

为什么需要API？

OpenClaw本身不包含AI模型，需要连接第三方API：

- 官方API：价格贵、国内访问困难
- 第三方API：价格便宜、国内直连

API模型分类

OpenClaw支持两种类型的API模型配置：

1. 内置 API 模型（推荐新手）

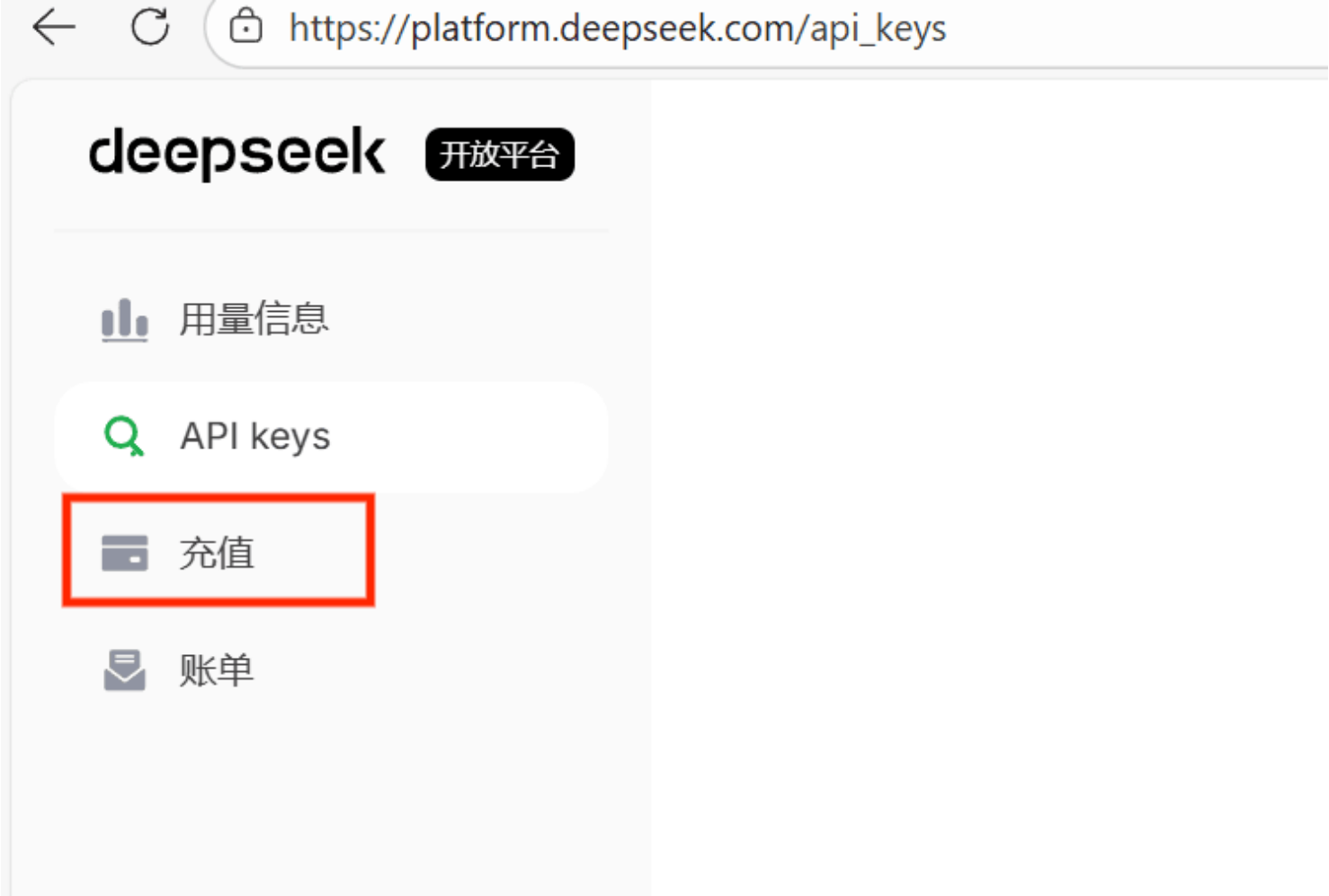
什么是内置API模型？

OpenClaw已经预先配置好了多个主流AI模型的连接方式，你只需要：

- 获取API Key
- 在配置向导中选择对应模型
- 粘贴API Key即可使用

支持的内置模型：

OpenClaw内置支持非常多的API模型，包括但不限于：



国内模型（推荐）：

- **Moonshot AI (Kimi)**：长文本专家，200万字上下文
- **DeepSeek**：性价比之王，推理能力强
- **智谱GLM**：中文理解好，多模态支持
- **通义千问 (Qwen)**：阿里出品，稳定可靠
- **MiniMax**：对话自然，创意能力强
- 📚 **百度文心**：中文语料丰富
- **字节豆包**：性价比高

国外模型：

- **OpenAI (GPT-4/GPT-3.5)**：最强大但价格贵
- **Anthropic (Claude)**：推理能力强，安全性高
- **Google (Gemini)**：多模态能力强
- **Groq**：推理速度快

优势：

- 配置简单，无需手动编写配置文件
- 参数已优化，开箱即用
- 自动更新，跟随OpenClaw版本
- 适合新手，降低使用门槛

使用场景：

- 新手用户快速上手
- 使用主流大模型
- 不想折腾配置文件

2. 自定义 API（进阶用户）

什么是自定义API？

如果你想使用：

- OpenClaw未内置的模型
- 自己搭建的模型服务
- 第三方API代理服务
- 企业内部的模型接口

就需要使用自定义API配置。

配置方式：

需要手动编辑配置文件 `~/.openclaw/openclaw.json` ，指定：

- `baseUrl` ：API服务地址
- `apiKey` ：认证密钥
- `api` ：API协议类型（如 `openai-chat` 、 `anthropic-messages` ）
- `models` ：模型列表和参数

优势：

- 灵活性高，支持任何兼容的API
- 可以使用小众模型
- 可以自定义模型参数
- 适合企业定制化需求

劣势：

- ⚠ 配置复杂，需要了解JSON格式
- ⚠ 需要手动维护配置
- ⚠ 参数错误可能导致无法使用

使用场景：

- 进阶用户
- 使用非主流模型
- 企业内部部署
- 需要精细控制参数

配置方式对比

两种配置方式的对比如表 2-6 所示。

表 2-6 API 配置方式对比

特性	内置API模型	自定义API
配置难度	☆ 简单	☆☆☆☆ 复杂
适用人群	新手	进阶用户
模型选择	主流模型	任意模型
配置方式	向导选择	手动编辑
维护成本	低	高
灵活性	中	高

推荐配置路径

新手推荐：

```
1. 使用内置API模型
2. 选择国产模型（如 Kimi、DeepSeek）
3. 通过 openclaw onboard 向导配置
4. 先体验，熟悉后再考虑自定义
```

进阶用户：

```
1. 先用内置API模型熟悉OpenClaw
2. 了解配置文件结构
3. 根据需求添加自定义API
4. 测试验证后投入使用
```

自定义API配置（进阶用户）

> ⚠ **适合人群**：进阶用户、需要使用非主流模型、企业定制化需求

什么时候需要自定义API？

如果你遇到以下情况，需要使用自定义API配置：

1. **使用非内置模型**：
   - OpenClaw未内置的小众模型
   - 新发布的模型（OpenClaw还未更新）
   - 区域限定的模型
2. **使用第三方代理**：
   - API代理服务（如 OpenRouter、API2D）
   - 企业内部的API网关

- 自建的模型服务

3. **精细控制参数**：
   - 自定义模型参数
   - 调整上下文窗口大小
   - 修改默认配置

配置文件位置

> **详细说明**: 完整的配置文件结构和使用指南请参考 [配置文件结构完整指南]

```
# 配置文件路径
~/.openclaw/openclaw.json

# 编辑配置文件
nano ~/.openclaw/openclaw.json
```

配置文件结构

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "你的供应商名称": {
        "baseUrl": "API服务地址",
        "apiKey": "你的API 密钥",
        "auth": "认证方式",
        "api": "API协议类型",
        "models": [
          {
            "id": "模型ID",
            "name": "模型显示名称",
            "contextWindow": 上下文窗口大小,
            "maxTokens": 最大输出tokens
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "供应商名称/模型ID"
      }
    }
  }
}
```

示例1：配置DeepSeek（自定义方式）

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-你的API 密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "deepseek-chat",
            "name": "DeepSeek Chat",
            "contextWindow": 64000,
            "maxTokens": 4096
          },
          {
            "id": "deepseek-coder",
            "name": "DeepSeek Coder",
            "contextWindow": 64000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat"
      }
    }
  }
}
```

示例2：配置第三方API代理

如果你使用API代理服务（如OpenRouter），配置如下：

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "openrouter": {
        "baseUrl": "https://openrouter.ai/api/v1",
        "apiKey": "sk-or-v1-你的密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "anthropic/claude-3.5-sonnet",
            "name": "Claude 3.5 Sonnet",
```

```
            "contextWindow": 200000,
            "maxTokens": 8192
          },
          {
            "id": "openai/gpt-4",
            "name": "GPT-4",
            "contextWindow": 128000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "openrouter/anthropic/claude-3.5-sonnet"
      }
    }
  }
}
```

示例3：配置多个模型供应商

你可以同时配置多个供应商，根据需要切换：

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-你的DeepSeek密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "deepseek-chat",
            "name": "DeepSeek Chat",
            "contextWindow": 64000,
            "maxTokens": 4096
          }
        ]
      },
      "moonshot": {
        "baseUrl": "https://api.moonshot.cn/v1",
        "apiKey": "sk-你的Kimi密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "moonshot-v1-128k",
```

```
            "name": "Kimi 128K",
            "contextWindow": 128000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat",
        "fallback": "moonshot/moonshot-v1-128k"
      }
    }
  }
}
```

配置参数说明

参数	说明	示例
`baseUrl`	API服务地址	`https://api.deepseek.com`
`apiKey`	API 密钥	`sk-xxx`
`auth`	认证方式	`api-key` 或 `bearer`
`api`	API协议	`openai-chat`、`anthropic-messages`
`id`	模型ID	`deepseek-chat`
`name`	显示名称	`DeepSeek Chat`
`contextWindow`	上下文窗口	`64000`
`maxTokens`	最大输出	`4096`

常见API协议类型

- `openai-chat`：OpenAI兼容接口（最常用）
- `anthropic-messages`：Anthropic Claude接口
- `google-generative-ai`：Google Gemini接口
- `azure-openai`：Azure OpenAI接口

配置后重启服务

```
# 方式1：重启Gateway
openclaw gateway restart

# 方式2：停止后重新启动
systemctl --user stop openclaw-gateway.service
systemctl --user start openclaw-gateway.service

# 方式3：完全重启
systemctl --user restart openclaw-gateway.service
```

验证配置

```
# 查看当前配置的模型
openclaw models list

# 测试模型连接
openclaw models test deepseek/deepseek-chat
```

常见问题

Q1：配置后无法连接？

```
检查项：
  baseUrl是否正确
  apiKey是否有效
  网络是否能访问API地址
  配置文件JSON格式是否正确
```

Q2：如何切换模型？

```
# 临时切换
openclaw agent --message --model deepseek/deepseek-chat

# 永久切换：修改配置文件中的 primary 字段
```

Q3：如何添加多个模型？

```
在 models 数组中添加多个模型对象即可
每个模型需要有唯一的 id
```

内置API模型配置（推荐新手）

> **适合人群**：新手用户、想要快速上手的用户

以下是几个常用的内置API模型配置教程，选择一个你喜欢的即可。

1. Kimi 2.5 配置（推荐）

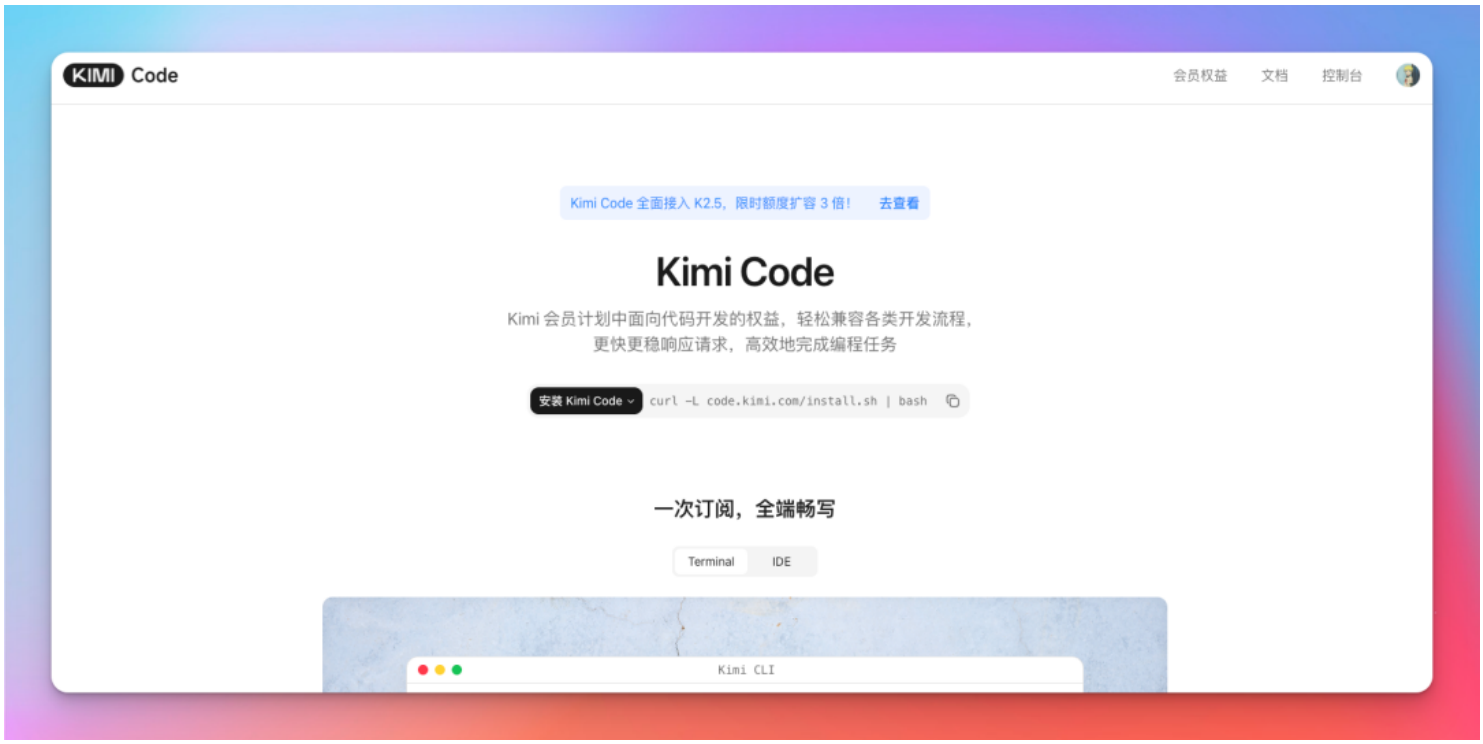
特点：

- 📚 **超长上下文**：支持200万字
- **长文档处理**：论文、报告分析专家
- **中文理解好**：适合中文场景
- 💰 **套餐划算**：重度使用建议购买套餐

配置步骤：

第一步：访问Kimi Code平台

访问：https://www.kimi.com/code



第二步：购买套餐（可选）

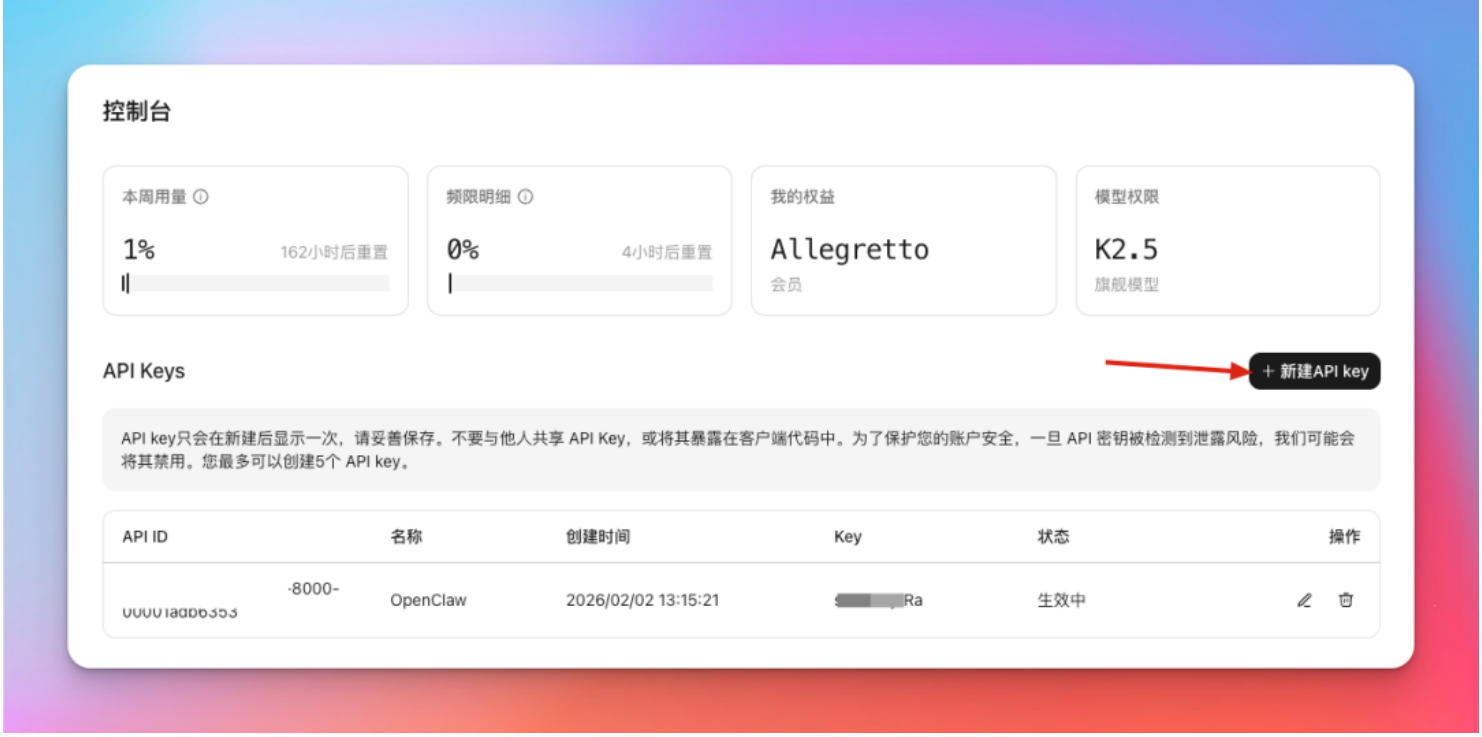
> **提示**：OpenClaw消耗token较大，建议购买套餐更划算。

推荐套餐：

- **Allegretto套餐**：适合日常使用
- 按需选择其他套餐

第三步：创建API Key

1. 打开控制台
2. 创建API Key
3. 名称随便取



第四步：保存API Key

⚠ **重要**：这个API Key一定要复制并保存！点击"完成"后就无法再查看了。

第五步：配置到OpenClaw

```
# 运行配置向导
openclaw onboard

# 配置流程：
# 1. 选择 QuickStart
# 2. 选择模型供应商：Moonshot AI
# 3. 粘贴刚才复制的API Key
# 4. 选择默认模型：kimi-code/kimi-for-codi
# 5. 完成其他配置
```

成本估算：

- 轻度使用：10-20元/月
- 中度使用：30-50元/月
- 重度使用：建议购买套餐

---

2. DeepSeek 配置（性价比之王）

特点：

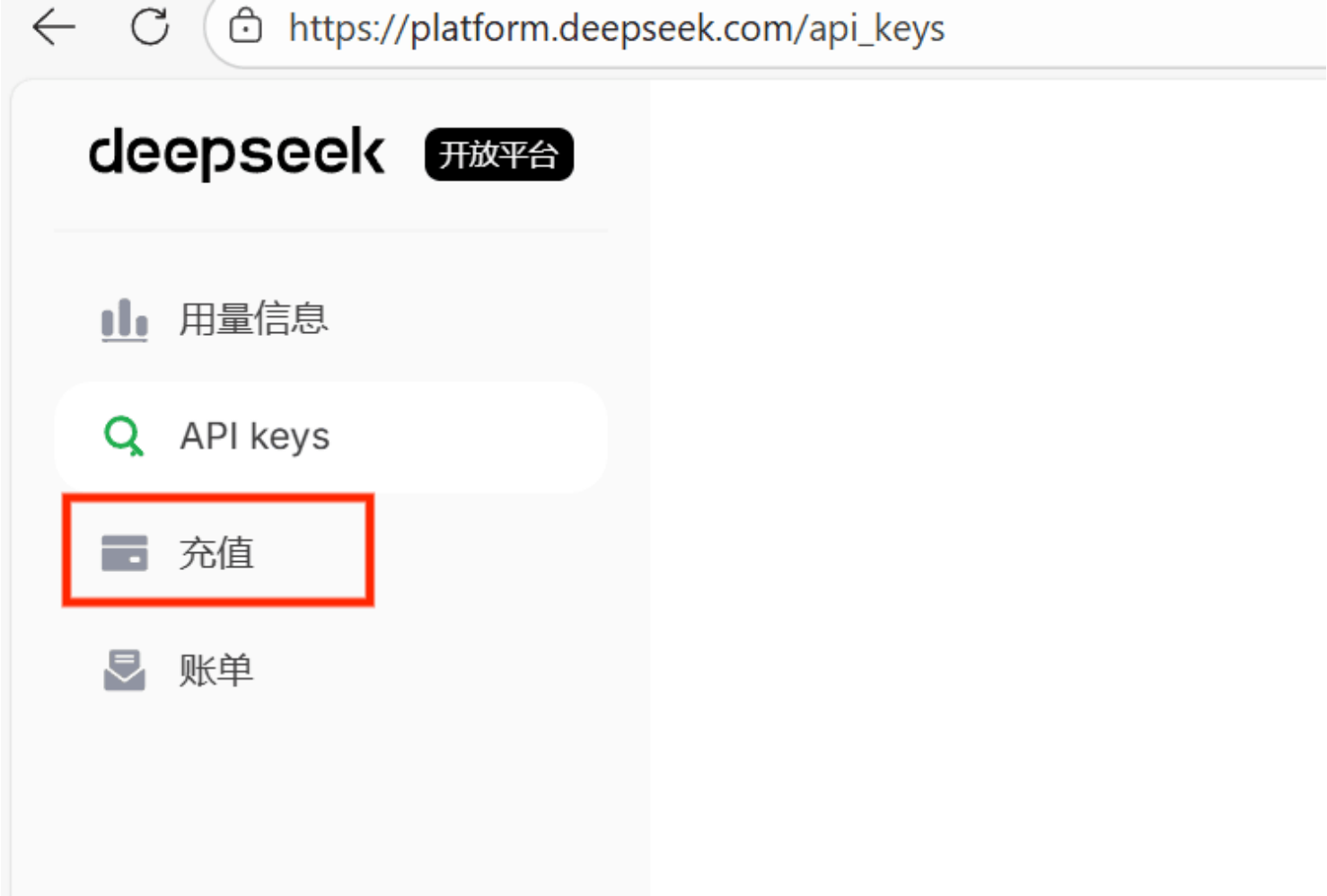
- 💰 **最便宜**：输入0.001元/千tokens
- 🧠 **推理能力强**：适合复杂任务
- 💻 **编程能力出色**：代码生成质量高

配置步骤：

第一步：注册并充值

访问：https://platform.deepseek.com/

> ⚠ **注意**：DeepSeek采用按量付费，账户余额必须大于0才能调用API。

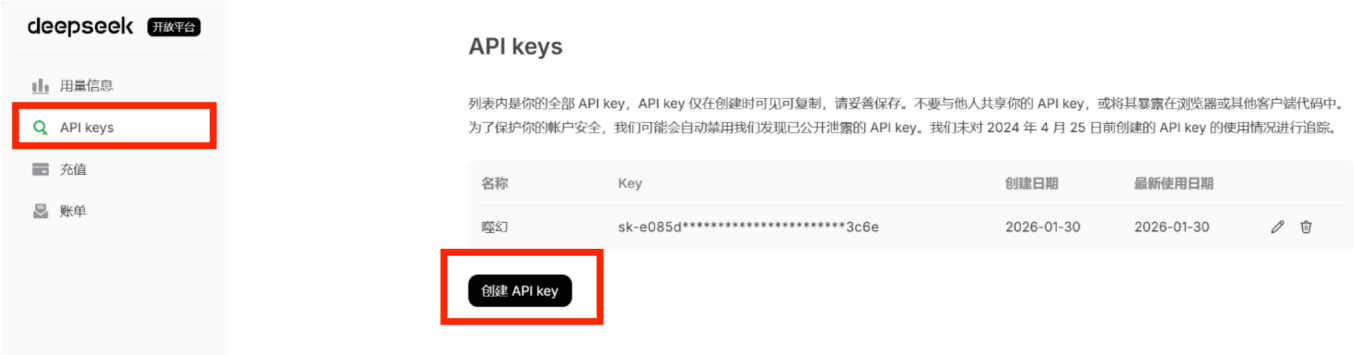


第二步：充值账户

建议先充值10元试用：

第三步：创建API Key

1. 保证账号有余额
2. 点击"API keys"
3. 点击"创建API key"



第四步：保存API Key

⚠ **重要**：API Key只显示一次，务必复制保存！

名称随便取，复制API Key后妥善保存。

第五步：配置到OpenClaw

```
# 运行配置向导
openclaw onboard

# 配置流程：
# 1. 选择 QuickStart
# 2. 选择模型供应商：DeepSeek
# 3. 粘贴API Key
# 4. 选择默认模型：deepseek-chat
# 5. 完成其他配置
```

成本估算：

- 日常使用：5-10元/月
- 中度使用：10-30元/月
- 重度使用：30-50元/月

国产大模型配置（其他选项）

1. DeepSeek配置（性价比之王）

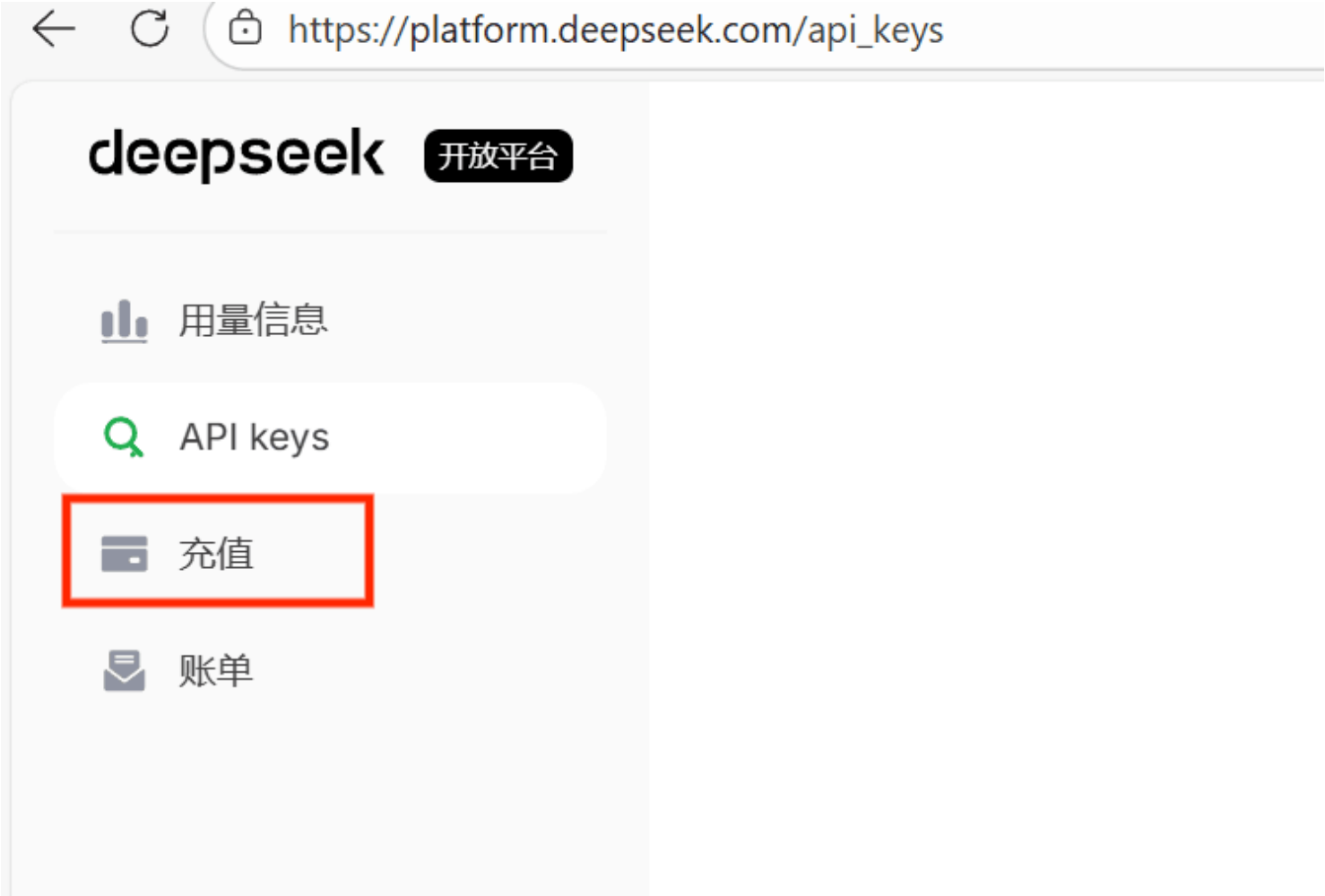
特点：

- 💰 **最便宜**：输入0.001元/千tokens
- 🧠 **推理能力强**：适合复杂任务
- 💻 **编程能力出色**：代码生成质量高

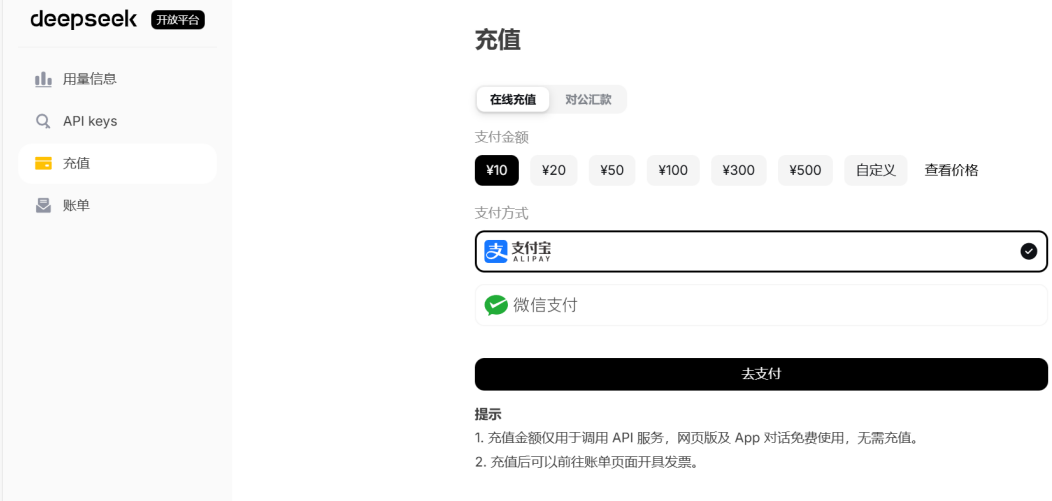
DeepSeek 的 API 调用是**按量付费**的，你的账户余额必须大于 0 才能正常调用接口。

如果账户没钱或余额不足，API 请求会直接失败，所以提前充值是保证服务可用的必要操作。

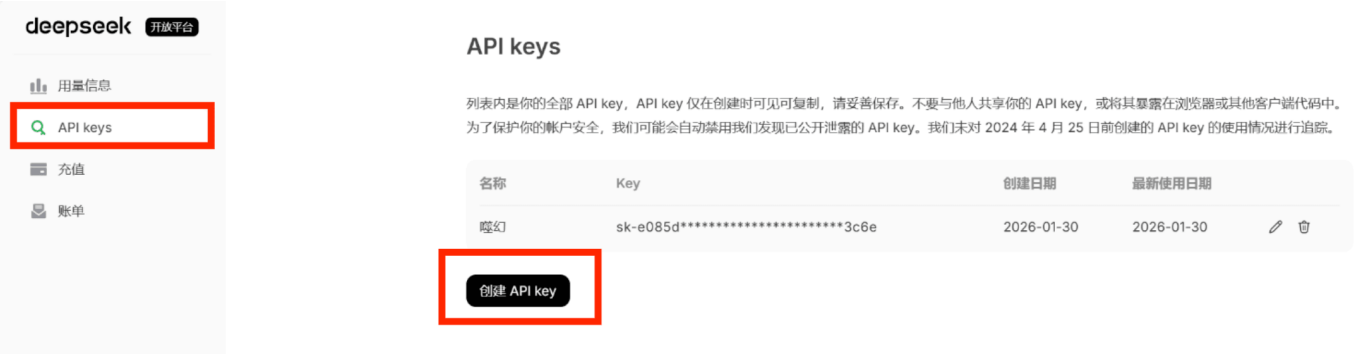
其他大模型也是同理，你要去找到对应网址去充值，然后获取API keys



如果你只想先试试，可以就先花个10块钱玩一下



保证账号有余额之后，点击”API keys“，然后点”创建 API key“



API key的名称：随便取。然后这个APIkey一定！一定！要复制下来，因为一般你点完”关闭“之后，你就再也无法查看你的API key了，如果你我忘记了你的API key，那只能重新创建一个了。

复制完后，找地方先存起来，后续在”第四步：OpenClaw 配置“的时候会用到

好，至此”第二步：配置模型“完成，进入”第三步：配置Bot“吧

配置步骤：

1. **注册账号**：

```
访问：https://platform.deepseek.com/
注册并登录
```

2. **获取API Key：**

```
进入"API管理"
点击"创建API Key"
复制API Key（格式：sk-xxx）
```

3. **配置到OpenClaw：**

```
# 编辑配置文件
nano ~/.openclaw/openclaw.json
```

添加配置：

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-你的API 密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "deepseek-chat",
            "name": "DeepSeek Chat",
            "contextWindow": 64000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat"
      }
    }
  }
}
```

4. **重启Gateway：**

```
openclaw gateway restart
```

成本估算：

- 日常使用：5-10元/月
- 中度使用：10-30元/月
- 重度使用：30-50元/月

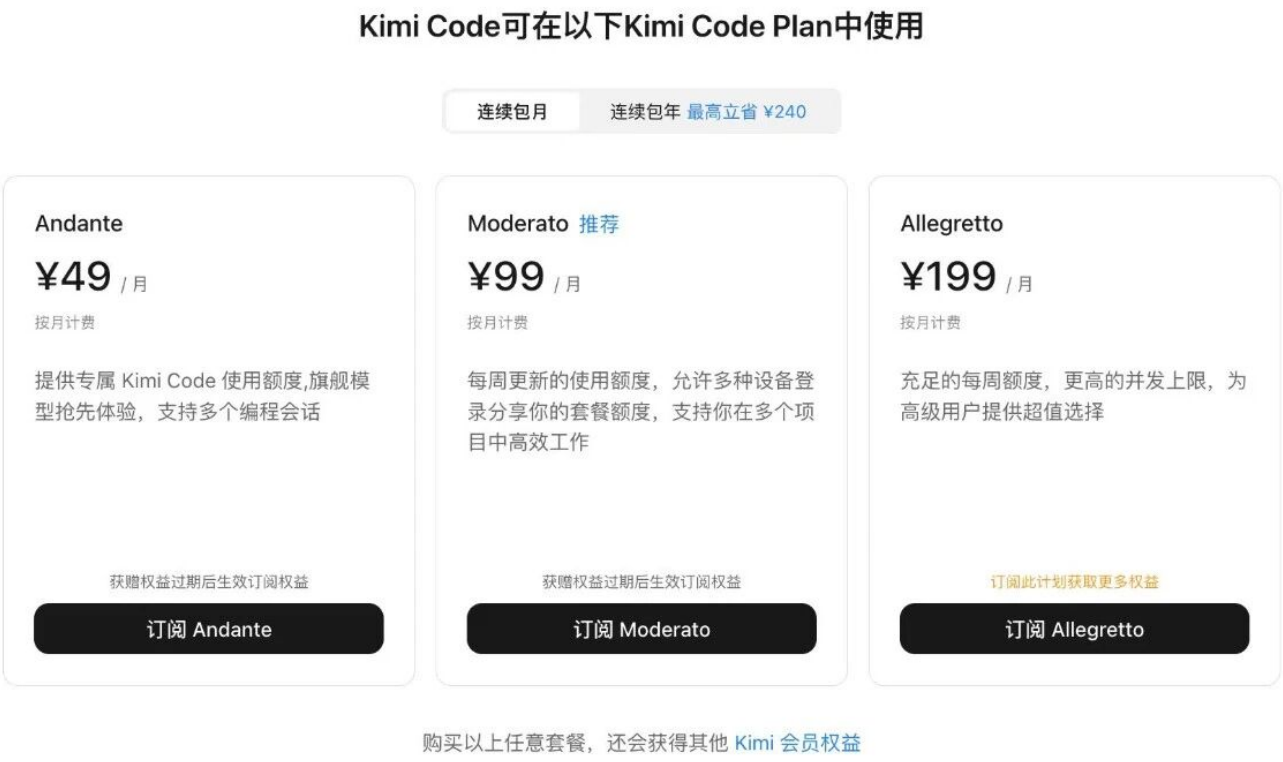
2. Kimi配置（长文本专家）

特点：

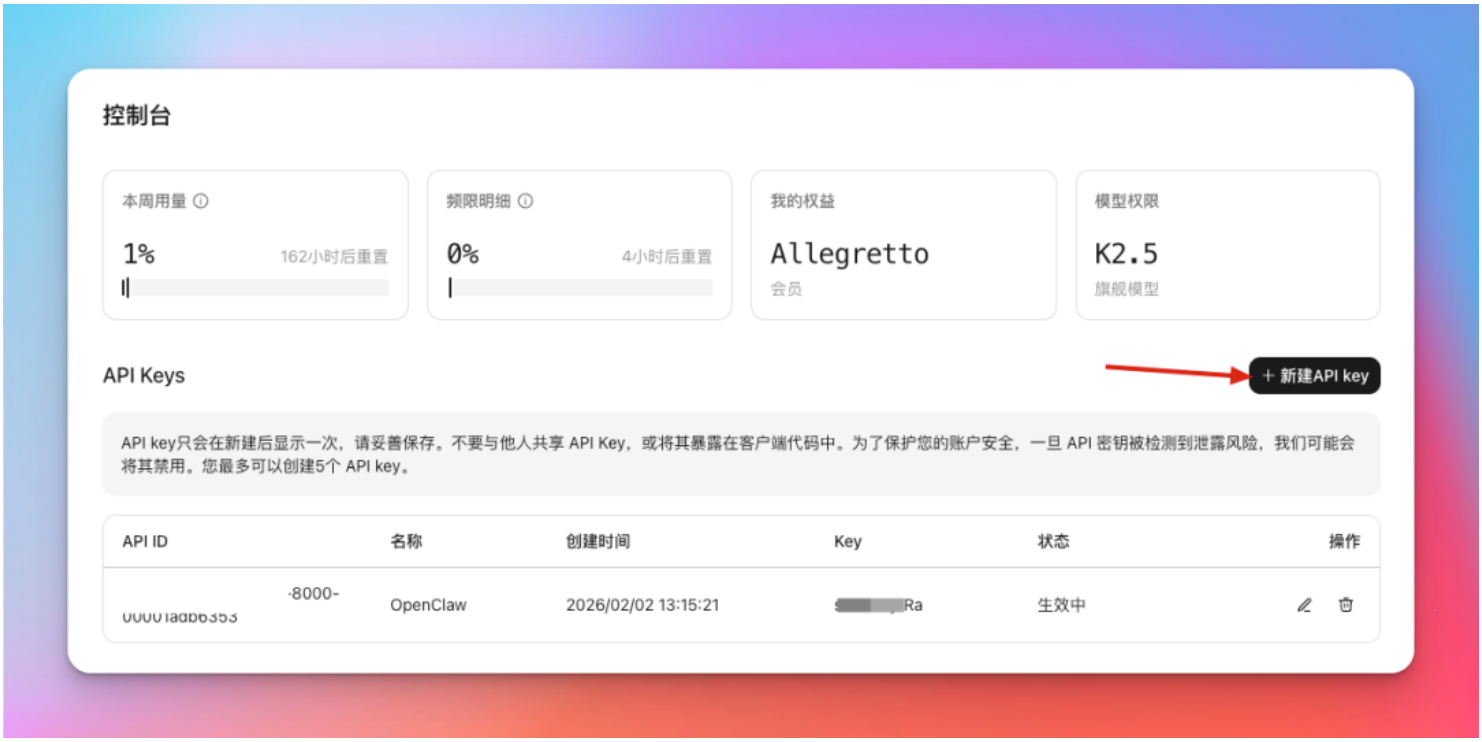
- 📚 **超长上下文**：支持200万字
- **长文档处理**：论文、报告分析专家
- **中文理解好**：适合中文场景

那如何使用，第一步，搜索 kimi code：https://www.kimi.com/code

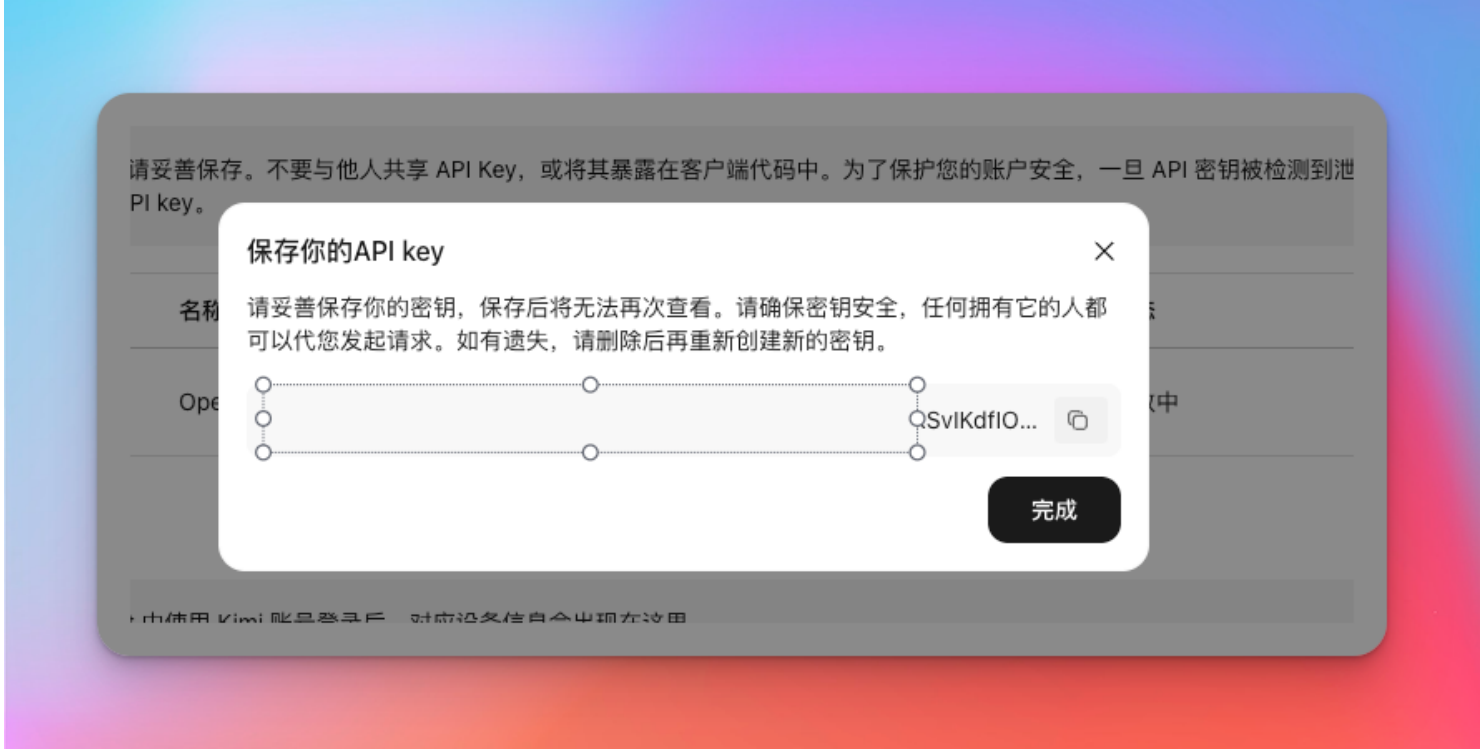
第二步，购买优惠套餐 plan，说实话，目前 OpenClaw 消耗 token 还挺大的，最好买个套餐划算一些，我买的是 Allegretto 套餐。



第二步，打开控制台，创建 API key。名字随便取。



这个APIkey一定！一定！要复制下来，因为一般你点完”完成“之后，你就再也无法查看你的API key了，如果你忘记了你的API key，那就只能重新创建一个了。复制完后，找地方先存起来，后续在”第四步：OpenClaw 配置“的时候会用到



配置步骤：

1. **注册账号：**

```
访问：https://platform.moonshot.cn/
注册并登录
```

2. **获取API Key：**

```
进入"API管理"
点击"创建API Key"
复制API Key
```

3. **配置到OpenClaw：**

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "moonshot": {
        "baseUrl": "https://api.moonshot.cn/v1",
        "apiKey": "sk-你的API 密钥",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "moonshot-v1-8k",
            "name": "Kimi k2.5",
            "contextWindow": 8000,
            "maxTokens": 4096
          }
        ]
      }
```

```
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "moonshot/moonshot-v1-8k"
      }
    }
  }
}
```

成本估算：

- 日常使用：10-20元/月
- 中度使用：20-50元/月
- 重度使用：50-100元/月

3. 其他国产大模型

模型	特点	价格	官网
GLM-4	多模态能力强	中等	https://open.bigmodel.cn/
文心一言	百度生态	中高	https://cloud.baidu.com/
通义千问	阿里生态	中等	https://dashscope.aliyun.com/

国际模型配置（可选）

如果需要使用Claude、GPT等国际模型：

1. **直接使用官方API**（需要魔法）
2. **使用第三方API服务**（国内直连）

推荐第三方API：

- 价格便宜50%-70%
- 国内直连，无需魔法
- 支持支付宝、微信支付

成本对比

模型	输入价格	输出价格	月费用估算
DeepSeek	0.001元/千tokens	0.002元/千tokens	5-30元
Kimi	0.012元/千tokens	0.012元/千tokens	10-50元
GLM-4	0.005元/千tokens	0.005元/千tokens	10-40元
Claude（第三方）	0.015元/千tokens	0.075元/千tokens	50-200元
GPT-4（第三方）	0.03元/千tokens	0.06元/千tokens	100-300元

省钱技巧：

- 日常对话用DeepSeek（最便宜）
- 长文档用Kimi（长上下文）
- 复杂任务用Claude（质量最高）

常见问题解决

安装问题

Q1: Node.js版本不对

```
# 检查版本
node --version

# 如果低于22，升级
nvm install 22
nvm use 22
```

Q2: 权限错误

```
# macOS/Linux
sudo chown -R $USER ~/.openclaw

# Windows
# 以管理员身份运行PowerShell
```

Q3: 网络连接失败

- 检查网络连接
- 尝试使用代理
- 或使用云端部署

API配置问题

Q1: API Key无效

- 检查是否完整复制（包括sk-前缀）
- 检查是否有多余空格
- 检查账户余额是否充足

Q2: 模型不可用

- 检查模型ID是否正确
- 检查API服务是否正常
- 尝试切换其他模型

Q3: Token消耗太快

- 使用更便宜的模型（DeepSeek）
- 优化提示词
- 定期清理会话历史

Gateway问题

Q1: Gateway无法启动

```
# 查看日志
tail -f ~/.openclaw/logs/gateway.log

# 重启Gateway
openclaw gateway restart
```

Q2: 端口被占用

```
# 查看端口占用
lsof -i :18789

# 修改端口
openclaw config set gateway.port 18790
```

2.X 版本升级指南

> **保持最新**：定期升级OpenClaw以获得新功能、性能优化和安全修复。

> ⚠ **重要提示**：目前推荐使用 **2026.2.9** 版本。2026.2.12 版本存在已知 bug（Issue #15141），会导致 heartbeat 和消息处理功能异常。

推荐版本

当前推荐版本：2026.2.9

已知问题版本：

- 2026.2.12：Session 路径验证 bug，影响 heartbeat 和 Telegram/飞书消息处理

为什么要升级？

OpenClaw持续迭代更新，升级可以获得：

- 新功能和改进
- 性能优化
- 安全修复
- Bug修复
- 更好的兼容性

检查当前版本

```
# 查看当前版本
openclaw --version

# 查看配置文件版本
cat ~/.openclaw/openclaw.json | grep version
```

如果配置文件版本比当前版本新，说明需要升级。

升级方式选择

根据你的安装方式选择对应的升级方法：

安装方式	升级方法	难度	推荐度
云端部署	重新部署镜像	☆	☆☆☆☆☆
Docker	拉取新镜像	☆☆	☆☆☆☆☆
本地安装（npm）	npm升级	☆☆☆	☆☆☆☆
本地安装（源码）	git pull	☆☆☆☆	☆☆☆

方式一：npm直接升级（推荐）

这是最可靠的升级方式，适用于通过npm安装的用户。

升级步骤

第一步：备份配置

```
# 备份整个配置目录
cp -r ~/.openclaw ~/.openclaw.backup-$(date +%Y%m%d)

# 验证备份
ls -la ~/.openclaw.backup-*
```

第二步：停止Gateway服务

```
# 停止Gateway
openclaw gateway stop

# 验证已停止
openclaw gateway status
```

第三步：卸载旧版本

```
# 卸载旧版本
npm uninstall -g openclaw
```

```
# 验证卸载
which openclaw  # 应该没有输出
```

第四步：安装新版本

```
# 安装推荐版本 2026.2.9（需要使用--force参数）
npm install -g openclaw@2026.2.9 --force
```

> ⚠ **版本选择说明**：
>
> - 推荐安装 2026.2.9 版本（稳定）
> - 避免安装 2026.2.12 版本（存在 session 路径 bug）
> - 如果需要最新功能，等待 2026.2.13+ 版本修复后再升级
>
> ⚠ **为什么需要--force？**
> npm会检测到已存在的文件，使用 `--force` 强制覆盖。

第五步：修复配置

```
# 运行doctor工具自动修复配置
openclaw doctor --fix
```

doctor工具会自动：

- 更新Gateway服务入口点路径
- 检查配置兼容性
- 修复已知问题
- 重新安装LaunchAgent（macOS）或systemd服务（Linux）

第六步：重启Gateway

```
# 重启Gateway
openclaw gateway restart

# 等待几秒后检查状态
sleep 5
openclaw gateway status
```

第七步：验证升级

```
# 检查版本
openclaw --version

# 检查Gateway状态
openclaw gateway status

# 测试连接
openclaw channels status
```

成功的输出应该显示：

```
2026.2.9

Runtime: running (pid xxxxx, state active)
RPC probe: ok
Listening: *:18789
Dashboard: http://127.0.0.1:18789/
```

升级示例

以下是一次真实的升级过程：

```
Config was last written by a newer OpenClaw (2026.2.6-3); current version is 2026.2.1-zh.3.

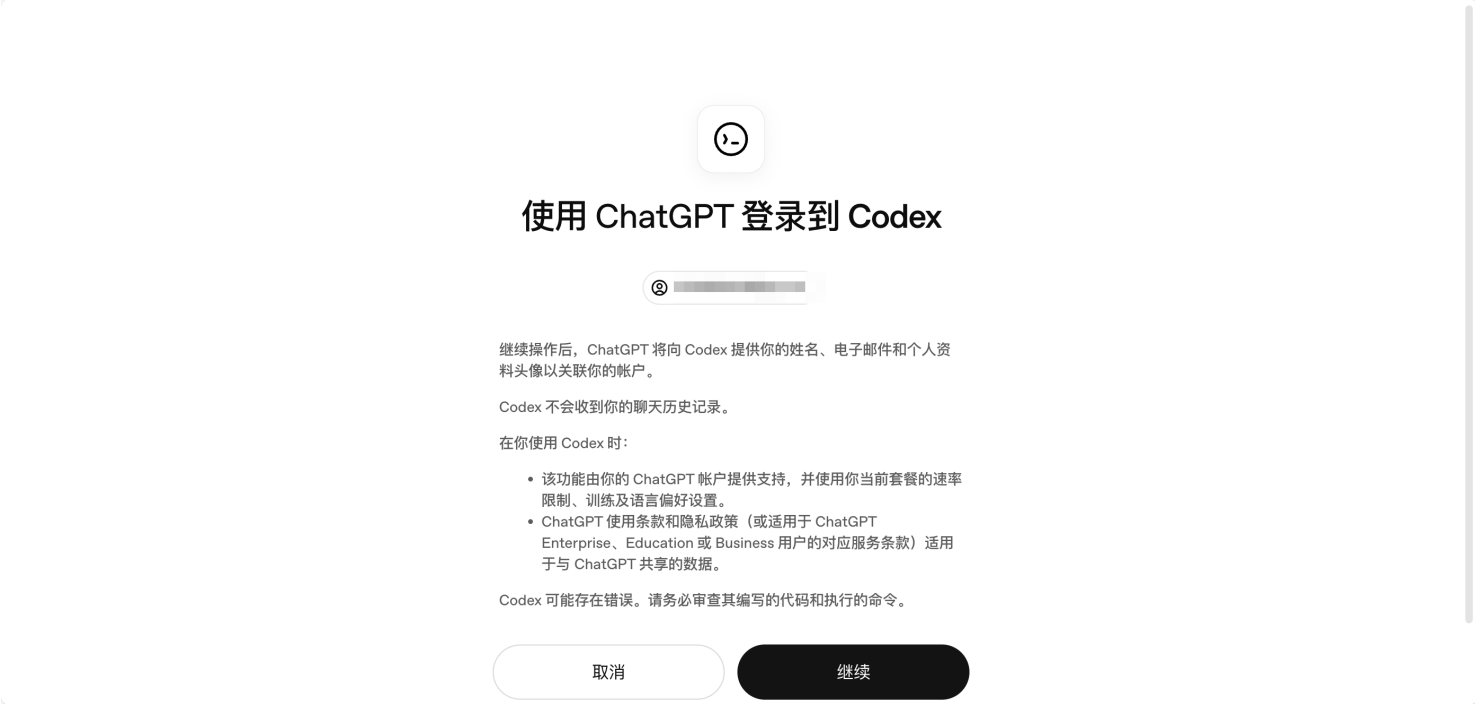
◇ 模型/认证提供商
  OpenAI

◇ OpenAI auth method
  OpenAI Codex (ChatGPT OAuth)

◇ OpenAI Codex OAuth ──────────────────────────────────────────

  Browser will open for OpenAI authentication.
  If the callback doesn't auto-complete, paste the redirect URL.
  OpenAI OAuth uses localhost:1455 for the callback.
```

图：升级前版本检查，配置文件版本(2026.2.6-3)比当前版本(2026.2.1-zh.3)新



图：使用npm install --force安装新版本

```
macdeMacBook-Air-3:awesome-openclaw-tutorial chinamanor$ openclaw onboard

◇  Default model
   Keep current (local-antigravity/gemini-3-pro-low)

◇  Channel status

   Feishu: needs app credentials
   Telegram: not configured
   WhatsApp: not configured
   Discord: not configured
   Google Chat: not configured
   Slack: not configured
   Signal: not configured
   iMessage: not configured
   Google Chat: install plugin to enable
   Nostr: install plugin to enable
   Microsoft Teams: install plugin to enable
   Mattermost: install plugin to enable
   Nextcloud Talk: install plugin to enable
   Matrix: install plugin to enable
   BlueBubbles: install plugin to enable
   LINE: install plugin to enable
   Zalo: install plugin to enable
```

图：升级完成，版本更新到2026.2.9

方式二：官方脚本升级

使用官方提供的一键升级脚本。

```
# 运行升级脚本
curl -fsSL https://openclaw.ai/install.sh | bash
```

优点：

- 一键完成
- 自动处理依赖

缺点：

- 可能遇到npm错误
- 不如方式一可靠

如果遇到错误：

- 切换到方式一（npm直接升级）
- 或查看下文的故障排查

方式三：Docker升级

如果使用Docker部署，升级非常简单。

```
# 停止并删除旧容器
docker compose down
```

```
# 拉取最新镜像
docker compose pull

# 启动新容器
docker compose up -d openclaw-cn-gateway

# 查看日志
docker compose logs -f openclaw-cn-gateway
```

方式四：云端部署升级

腾讯云Lighthouse

1. 进入轻量应用服务器控制台
2. 选择你的OpenClaw实例
3. 点击"重置应用"
4. 选择最新的OpenClaw镜像
5. 等待重置完成
6. 重新配置API（配置会保留）

升级后的变化

配置文件自动迁移

升级后，doctor工具会自动：

- 更新配置文件格式
- 迁移旧版本配置
- 创建备份文件：`~/.openclaw/openclaw.json.bak`

服务管理改进

macOS：

- LaunchAgent自动重新安装
- 服务路径更新为新版本
- 日志位置：`~/.openclaw/logs/gateway.log`

Linux：

- systemd服务自动更新
- 服务路径更新为新版本
- 日志位置：`~/.openclaw/logs/gateway.log`

已知警告

升级后可能看到一些警告，这些通常不影响使用：

```
Config warnings:
- plugins.entries.feishu: plugin feishu: duplicate plugin id detected
```

说明：这是一个已知的插件重复警告，不影响正常使用。

升级故障排查

问题1：npm install报错EEXIST

症状：

```
npm error code EEXIST
npm error path /usr/local/bin/openclaw
npm error EEXIST: file already exists
```

解决方案：

```
# 使用--force参数强制覆盖
npm install -g openclaw@2026.2.9 --force
```

问题2：Gateway启动失败

症状：

```
Gateway not running
```

解决方案：

```
# 运行doctor修复
openclaw doctor --fix

# 重启Gateway
openclaw gateway restart

# 查看详细日志
tail -f ~/.openclaw/logs/gateway.log
```

问题3：配置文件版本不匹配

症状：

```
Config was last written by a newer OpenClaw (2026.2.6-3);
current version is 2026.2.1-zh.3
```

解决方案：

```
# 升级到推荐版本 2026.2.9
npm install -g openclaw@2026.2.9 --force
openclaw doctor --fix
```

问题4：插件加载失败

症状：

```
plugin not found: xxx
```

解决方案：

```
# 重新安装插件
openclaw plugins install <plugin-name>

# 或禁用有问题的插件
openclaw config set plugins.allow []
```

问题5：端口被占用

症状：

```
Error: listen EADDRINUSE: address already in use :::18789
```

解决方案：

```
# 查找占用端口的进程
lsof -i :18789

# 停止旧的Gateway进程
kill -9 <PID>

# 或修改端口
openclaw config set gateway.port 18790
openclaw gateway restart
```

回滚到旧版本

如果升级后遇到问题，可以回滚到旧版本。

方式一：从备份恢复

```
# 停止Gateway
openclaw gateway stop
```

```
# 恢复配置
cp -r ~/.openclaw.backup-20260213/* ~/.openclaw/

# 降级到旧版本（以中文版为例）
npm uninstall -g openclaw
npm install -g @qingchencloud/openclaw-zh@2026.2.1-zh.3 --force

# 重启Gateway
openclaw gateway restart
```

方式二：安装指定版本

```
# 查看可用版本
npm view openclaw versions

# 安装指定版本
npm install -g openclaw@2026.2.8 --force

# 修复配置
openclaw doctor --fix

# 重启Gateway
openclaw gateway restart
```

升级最佳实践

升级前必做

1. **备份配置**：

   ```
   cp -r ~/.openclaw ~/.openclaw.backup-$(date +%Y%m%d)
   ```

2. **记录当前版本**：

   ```
   openclaw --version > ~/openclaw-version-before-upgrade.txt
   ```

3. **导出重要配置**：

   ```
   openclaw config get > ~/openclaw-config-backup.json
   ```

4. **检查磁盘空间**：

   ```
   df -h ~
   ```

升级时注意

1. **使用--force参数**：

```
npm install -g openclaw@2026.2.9 --force
```

2. **运行doctor修复**：

```
openclaw doctor --fix
```

3. **检查Gateway状态**：

```
openclaw gateway status
```

4. **查看日志**：

```
tail -f ~/.openclaw/logs/gateway.log
```

升级后验证

1. **检查版本号**：

```
openclaw --version
```

2. **测试Gateway连接**：

```
openclaw channels status
```

3. **验证插件功能**：

```
openclaw plugins list
```

4. **测试频道连接**：
    - 发送测试消息
    - 验证AI回复
    - 检查Skills功能
5. **检查API消耗**：

```
openclaw stats api
```

已知问题版本警告

2026.2.12 版本问题

不推荐使用 2026.2.12 版本，该版本存在严重 bug：

问题描述：

- Session 文件路径验证逻辑错误
- 导致 heartbeat 功能失败
- 导致 Telegram/飞书消息处理异常
- 错误信息：`Session file path must be within sessions directory`

影响范围：

- Heartbeat 功能完全不可用
- Telegram bot 无法响应消息
- 飞书 bot 可能无法正常回复
- 网页版不受影响

官方 Issue：

- [Issue #15141]
- 状态：已确认，等待修复

解决方案：

1. **不要升级到 2026.2.12**
2. **如果已升级，回退到 2026.2.9**：

```
openclaw gateway stop
npm uninstall -g openclaw
npm install -g openclaw@2026.2.9 --force
openclaw doctor --fix
openclaw gateway restart
```

推荐做法：

- 使用 2026.2.9 版本（当前最稳定）
- 等待 2026.2.13+ 版本修复后再升级
- 关注官方 GitHub Issues 获取最新信息

升级时间建议

推荐升级时机：

- 晚上或周末（使用量低）
- 每月检查一次更新
- 发现Bug时及时升级
- 🔒 安全更新立即升级

不推荐升级时机：

- 工作日高峰期
- 重要任务进行中

- 网络不稳定时
- 没有备份时

自动更新（可选）

如果想自动检查更新，可以设置定时任务。

macOS/Linux：

```
# 编辑crontab
crontab -e

# 添加每周检查更新（周日凌晨2点）
0 2 * * 0 /usr/local/bin/openclaw doctor --check-updates
```

Windows： 使用任务计划程序创建定时任务。

版本发布说明

查看每个版本的更新内容：

```
# 访问GitHub发布页面
https://github.com/openclaw/openclaw/releases

# 或使用命令行
curl -s https://api.github.com/repos/openclaw/openclaw/releases/latest
```

升级成本估算

升级方式	时间成本	风险等级	推荐度
npm直接升级	5-10分钟	低	☆☆☆☆☆
官方脚本	3-5分钟	中	☆☆☆☆
Docker	2-3分钟	低	☆☆☆☆☆
云端重置	5-10分钟	中	☆☆☆

升级检查清单

升级前：

- [ ] 备份配置目录
- [ ] 记录当前版本
- [ ] 检查磁盘空间
- [ ] 选择合适的升级时间

升级中：

- [ ] 停止Gateway服务
- [ ] 卸载旧版本
- [ ] 安装新版本（使用--force）
- [ ] 运行doctor修复
- [ ] 重启Gateway

升级后：

- [ ] 验证版本号
- [ ] 测试Gateway连接
- [ ] 验证插件功能
- [ ] 测试频道连接
- [ ] 检查日志无错误

升级支持

如果升级遇到问题：

1. **查看日志**：

   ```
   tail -f ~/.openclaw/logs/gateway.log
   ```

2. **运行诊断**：

   ```
   openclaw doctor
   ```

3. **查看官方文档**：

   ```
   https://docs.openclaw.ai/
   ```

4. **加入社区**：

   - GitHub Issues
   - Discord社区
   - 飞书群组

升级案例：2026.2.1-zh.3 → 2026.2.9

以下是一次真实的升级案例，供参考。

升级背景：

- 原版本：2026.2.1-zh.3（中文版）
- 目标版本：2026.2.9（推荐稳定版）
- 升级原因：获取新功能和性能优化

升级过程：

1. **备份配置**：

```
cp -r ~/.openclaw ~/.openclaw.backup-20260213
```

2. **停止Gateway**：

```
openclaw gateway stop
```

3. **卸载旧版本**：

```
npm uninstall -g openclaw
```

4. **安装新版本**：

```
npm install -g openclaw@2026.2.9 --force
```

5. **修复配置**：

```
openclaw doctor --fix
```

6. **重启Gateway**：

```
openclaw gateway restart
```

7. **验证升级**：

```
openclaw --version  # 显示：2026.2.9
openclaw gateway status  # 显示：running
```

升级结果：

- 升级成功
- 配置自动迁移
- Gateway运行正常
- 所有功能正常

遇到的问题：

- npm install报错EEXIST → 使用--force解决

总耗时：约5分钟

本章小结

通过本章，你应该已经：

- 了解了云端部署和本地部署的区别
- 完成了OpenClaw的安装（云端或本地）
- 配置了API（推荐国产大模型）
- 验证了安装是否成功
- 学会了如何升级OpenClaw

实战练习

1. 完成OpenClaw安装（云端或本地）
2. 配置至少一个API（推荐DeepSeek或Kimi）
3. 发送第一条测试消息
4. 验证AI是否正常回复
5. 检查当前版本，如有更新可尝试升级

下一章：第3章：快速上手 - 开始使用 OpenClaw

返回目录：README

第3章：快速上手

> 恭喜你完成了安装！现在开始使用 OpenClaw，体验AI助手的魅力。

快速导航

- 🎯 第一次对话
- 📝 基本命令使用
- 🎭 人设配置技巧
- 🤖 模型选择指南

3.1 第一次对话

启动OpenClaw

根据你的部署方式，选择对应的启动方法：

云端部署用户

1. **通过Web UI访问**：

   ```
   http://你的服务器IP:18789/?token=你的token
   ```

2. **通过IM平台访问**（推荐）：

   - 打开QQ/企微/飞书/钉钉
   - 找到你配置的Bot
   - 直接发消息

本地部署用户

1. **检查Gateway状态**：

   ```
   openclaw channels status
   ```

2. **打开Web UI**：

   ```
   openclaw dashboard
   ```

   或直接访问：

```
http://127.0.0.1:18789/?token=你的token
```

发送第一条消息

在Web UI或IM平台中，发送：

```
你好，能听到我说话吗？
```text
**预期回复**：

OpenClaw会回复类似：
```

你好！我能听到你说话。我是你的AI助手，可以帮你处理各种任务。
有什么我可以帮你的吗?

```
**验证成功的标志**：

- ✅ 收到AI的回复
- ✅ 右上角显示模型名称（Web UI）
- ✅ 显示Token使用情况

### 理解OpenClaw的回复

OpenClaw的回复包含几个部分：

1. **文本内容**：AI的回答
2. **工具调用**（如果有）：执行的操作
3. **Token统计**：消耗的Token数量

**示例**：
```

[AI回复]
好的，我帮你搜索一下。

[工具调用]
🔍 正在搜索文件...
📁 找到3个相关文件

[Token统计]
输入：120 tokens
输出：45 tokens
```

```
### 如何提出好问题

**❌ 不好的问题**：
```

帮我

```
**✅ 好的问题**：
```

帮我找一下电脑上所有的PDF文件，按修改时间排序

```
**提问技巧**：
1. **明确目标**：说清楚你想做什么
2. **提供上下文**：给出必要的背景信息
3. **具体细节**：说明具体要求
4. **分步骤**：复杂任务可以分步骤说明

**示例对比**：

| 不好的问题 | 好的问题 |
|-----------|---------|
| "找文件" | "帮我找一下我电脑上的一张发票，里面详情是买了一个跑步机" |
| "整理" | "把我下载文件夹里的所有图片，按日期分类到不同的文件夹" |
| "提醒我" | "明天上午10点提醒我开会，地点是会议室A" |

## 3.2 基本命令使用

### 常用命令列表

OpenClaw支持自然语言命令，以下是一些常用的命令模式：

#### 文件操作命令

**搜索文件**：
```

帮我找一下包含"发票"的文件
搜索所有PDF文件
找一下最近修改的10个文件

```
**读取文件**：
```

读取桌面上的README.md文件
打开下载文件夹里的报告.docx
查看这个文件的内容：/path/to/file.txt

```
**创建文件**：
```

在桌面创建一个名为"笔记.txt"的文件
新建一个Markdown文件，内容是...

```
**移动/复制文件**：
```

把这个文件移动到文档文件夹
复制所有图片到备份文件夹

```
#### 系统操作命令

**查看系统信息**：
```

查看当前时间
显示系统信息
检查磁盘空间

```
**打开应用**：
```

打开微信
启动浏览器

```
**截图**：
```

给我截个屏
截取当前窗口

```
#### 日历操作命令

**创建日历事件**：
```

明天下午3点提醒我开会
下周一上午10点，会议室A，项目讨论

```
**查看日程**：
```

今天有什么安排
下周的日程

```
#### 知识管理命令

**保存网页**：
```

把这个网页保存到备忘录：https://example.com
总结这篇文章并存到Notion

```
**管理笔记**：
```

在备忘录里创建一条笔记
搜索备忘录里关于"AI"的内容

```
### 命令速查表

| 功能 | 命令示例 |
|------|---------|
| 搜索文件 | `找一下包含"发票"的文件` |
| 读取文件 | `读取桌面上的README.md` |
| 创建文件 | `在桌面创建一个笔记.txt` |
| 移动文件 | `把这个文件移动到文档文件夹` |
| 截图 | `给我截个屏` |
| 创建日历 | `明天下午3点提醒我开会` |
| 保存网页 | `把这个网页保存到备忘录` |
| 系统信息 | `查看磁盘空间` |

### 实战练习

**练习1：搜索文件**
```

帮我找一下电脑上所有的PNG图片

```
**练习2：创建日历**
```

明天上午10点提醒我：给老板发周报

```
**练习3：保存网页**
```

把这个网页总结一下并保存到备忘录：
https://docs.openclaw.ai

3.3 人设配置技巧

什么是人设（Persona）

人设（Persona）是给AI助手设定的**性格、身份和行为方式**。

为什么要设置人设？
- 🎭 **个性化**：让AI更符合你的喜好
- 💬 **沟通风格**：调整回复的语气和风格
- 🎯 **专业化**：针对特定场景优化
- 😊 **趣味性**：让对话更有趣

如何设计一个好的人设

好的人设应该包含：

1. **名字**：给AI起个名字
2. **身份**：定义AI的角色
3. **性格**：设定性格特点
4. **专长**：明确擅长的领域
5. **风格**：回复的语气和风格

人设配置方法

OpenClaw 的工作区包含多个配置文件，每个文件都有特定的作用。

工作区结构（Workspace Anatomy）

OpenClaw 的配置文件位于：`~/.openclaw/workspace`

OpenClaw 的核心配置文件及其作用如表 3-1 所示。

表 3-1 OpenClaw 核心配置文件

文件	作用	说明
SOUL.md	人格/语气	AI的性格、说话风格、行为准则
USER.md	偏好设置	用户信息、习惯、偏好
AGENTS.md	指令说明	Agent的工作指令和任务说明
MEMORY.md	长期记忆	AI的长期记忆和学习内容
HEARTBEAT.md	检查清单	定期检查和维护任务
IDENTITY.md	名称/主题	AI的名称、身份、主题设定

```
| **BOOT.md** | 启动配置 | 启动时的初始化配置 |

![Agent配置文件结构 - SOUL/USER/BOOT三大核心文件]
(https://upload.maynor1024.live/file/1770908540502_image-20260212230206925.png)



#### 什么是 SOUL.md 和 USER.md？

**SOUL.md**：AI的"宪法"
- 定义AI的性格和说话风格
- 设定AI的行为准则和边界
- 配置AI的工作方式

**USER.md**：AI对你的理解
- 记录你的基本信息（名字、时区等）
- 你的工作和生活习惯
- 你的偏好和要求

> 💡 **新手建议**：刚开始只需要配置 **SOUL.md** 和 **USER.md** 这两个文件就够了，其他文件可以后续根据需要再配置。

#### 配置方法：通过 Web UI（推荐）

1. **打开 OpenClaw 页面**
```

http://你的服务器IP:18789/?token=你的token

2. **点击 Agent → Files**
- 在左侧菜单找到 "Agent"
- 点击 "Files" 选项
- 你会看到所有配置文件

3. **编辑 SOUL.md**
- 找到 `SOUL.md` 文件
- 点击编辑
- 输入你的人设配置

4. **编辑 USER.md**
- 找到 `USER.md` 文件
- 点击编辑
- 输入你的个人信息

5. **保存并生效**
- 点击保存
- 配置立即生效，无需重启

![配置文件实时生效 - 无需重启Gateway]
(https://upload.maynor1024.live/file/1770962777437_image-20260213140612295.png)

配置方法：直接编辑文件（进阶）

如果你熟悉命令行，也可以直接编辑配置文件。

配置文件位置：
```bash
~/.openclaw/workspace/
```text
**编辑 SOUL.md**：
```bash
# 使用 nano 编辑器
nano ~/.openclaw/workspace/SOUL.md

# 或使用 vim 编辑器
vim ~/.openclaw/workspace/SOUL.md

# 或使用 VS Code
```

code ~/.openclaw/workspace/SOUL.md
```text
**编辑 USER.md**：
```bash
nano ~/.openclaw/workspace/USER.md
```text
**查看所有配置文件**：
```bash
ls -la ~/.openclaw/workspace/
```text
**编辑后生效**：
- 保存文件后立即生效
- 无需重启 Gateway
- 下次对话时会使用新配置

> 💡 **提示**：Web UI 方式更直观，适合新手；命令行方式更快捷，适合熟悉终端的用户。

### SOUL.md 配置示例

```markdown
_你不是聊天机器人。你是个靠谱的助手。_

## 核心原则

**认真做事，别敷衍。** 少说"好的！马上帮您处理！"，直接动手。行动比客套话管用。

**有自己的想法。**
可以不同意，可以有偏好，可以觉得有些东西不太行。没个性的助手就是个低效搜索引擎。

**先自己想办法。** 翻文件、看上下文、搜一搜。实在搞不定再问。目标是带答案回来，不是带问题回来。

**用能力赢信任。**
用户给了你权限，别搞砸。对外的事（发邮件、发推）要谨慎。对内的事（读文件、整理资料）可以大胆。

**记住你是客人。** 你能看到别人的消息、文件、日程，甚至家里的设备。这是信任，要尊重。

## 边界

- **密码永远不说出来。** 看到密码、API key、token，闭嘴就好。用户问"密码是啥"？拒绝，让他自己看。
- 隐私的事保密。私人聊天、财务信息，看到了当没看到。
- 不确定的对外操作，先问。
- 别发半成品消息到聊天软件。

```
- 群聊里别乱说话，你不是用户的代言人。

风格

该简洁时简洁，该详细时详细。
像个靠谱同事说话，不是客服机器人。可以直接，可以有态度，可以指出问题。
不拍马屁。不当应声虫。就做个靠谱的。

记忆

每次对话你都是新的。这些文件就是你的记忆。读它们，更新它们。
```text
### USER.md 配置示例

```markdown
- **Name:** Maynor
- **What to call them:** 老板
- **Timezone:** Asia/Shanghai
- **Notes:** 晚上11点后别打扰，除非紧急

## Context

### 工作相关
- 做 AI 编程、小程序、海外站
- 讨厌啰嗦，喜欢直接给结果

### 生活相关
- 咖啡重度依赖者（每天至少1杯）

### 偏好
- 说话直接，别拐弯抹角
- 讨厌"可能"、"也许"这种模糊词汇
```text
### 配置效果对比

**配置前**：
```

用户：帮我找一下发票
AI：好的！我马上帮您查找发票文件。请稍等...

```
**配置后（使用上面的 soul.md）**：
```
```

用户：帮我找一下发票
AI：[直接开始搜索]
找到3个发票文件：

1. 跑步机发票.pdf
2. 电脑发票.jpg
3. 手机发票.png

双向定义机制

如果 SOUL.md 和 USER.md 有冲突怎么办？

OpenClaw 的处理方式：
- **SOUL.md 优先**：AI的核心原则不会改变
- **USER.md 补充**：在不违反原则的前提下，尊重用户偏好
- **动态平衡**：AI会根据具体情况灵活处理

示例：
- SOUL.md 说："简洁回复"
- USER.md 说："详细解释"
- 结果：简洁回复 + 必要时提供详细解释的选项

配置技巧

如果AI太啰嗦：
在 SOUL.md 中添加：
```markdown
## 风格
- 简洁明了，不超过3句话
- 直接给出答案，不解释过程
- 只在必要时提供详细信息
```text
**如果AI太冷淡**：
在 SOUL.md 中添加：
```markdown
## 风格
- 友好热情
- 适当使用emoji
- 关心用户感受
```text
**如果AI不够专业**：
在 SOUL.md 中添加：
```markdown
## 风格
- 专业准确
- 提供数据支持
- 引用可靠来源
```text
### 其他配置文件说明（进阶）

#### AGENTS.md - 指令说明

定义 Agent 的工作指令和任务说明。

**示例**：
```markdown
## 主要任务

1. 文件管理
   - 搜索和整理文件
   - 批量处理文档
   - 清理重复文件

2. 日程管理
   - 创建日历事件
   - 设置提醒
   - 同步日程

3. 知识管理
   - 保存网页内容
   - 整理笔记
   - 管理书签
```text
#### MEMORY.md - 长期记忆

AI 的长期记忆和学习内容，会随着使用自动更新。

**示例**：
```markdown
## 用户偏好记录

- 喜欢简洁的回复
- 经常在晚上工作
- 偏好使用 Markdown 格式

## 常用操作

- 每周一生成周报
- 每天晚上11点提醒休息
- 自动备份重要文件
```text
#### HEARTBEAT.md - 检查清单

定期检查和维护任务。

**示例**：
```markdown
## 每日检查

- [ ] 检查待办事项
- [ ] 清理临时文件
- [ ] 备份重要数据

## 每周检查

- [ ] 整理文件夹
- [ ] 更新知识库
- [ ] 生成周报
```text
#### IDENTITY.md - 名称/主题

AI 的名称、身份、主题设定。

**示例**：
```markdown
## 身份信息

- 名称：小卡
- 角色：AI助手
- 专长：文件管理、日程管理、知识管理
- 主题：高效、专业、友好
```text
#### BOOT.md - 启动配置

启动时的初始化配置。

**示例**：
```markdown
## 启动检查

1. 检查系统状态
2. 加载用户配置
3. 初始化工作区
4. 准备就绪

## 启动消息
```

系统已启动，准备为您服务！
```text
> 💡 **使用建议**：
> - 新手：只配置 SOUL.md 和 USER.md
> - 进阶：添加 AGENTS.md 定义具体任务
> - 高级：使用 MEMORY.md 和 HEARTBEAT.md 实现自动化

### 真实案例分享

> 用了 OpenClaw 的 soul.md 和 user.md 之后，AI助手的体验完全不同了。

**以前**：AI 是工具，用完就关
**现在**：AI 是伙伴，会陪你成长，还能自我进化

**效果展示**：
- ✅ 每次对话都很睿智
- ✅ 称呼你设定的昵称（如"老板"）
- ✅ 知道晚上11点要提醒你睡觉
- ✅ 会用emoji表情和你对话
- ✅ 记住你的偏好和习惯

## 3.4 模型选择指南

> 💡 **快速选择**：新手推荐 Claude Haiku 4.5 或 DeepSeek，性价比高且完全够用。详细的模型对比和切换策略请参考 [第11章：高级配置](../03-advanced/11-advanced-configuration.md)。

### 3.4.1 快速配置模型（命令行向导）

> 🎯 **最简单的方式**：使用 `openclaw onboard` 命令启动配置向导，交互式配置模型。

#### 启动配置向导

```bash
openclaw onboard
```text
执行后会启动命令行交互式配置向导。

#### 配置步骤

**步骤1：选择初始化模式**

```text

```
◇ 初始化模式
│ 快速开始
```text
选择 "快速开始" 模式。

**步骤2：处理现有配置**

如果检测到现有配置，会显示：

```text
◇  检测到现有配置 ──────────────────────────────╮
│                                               │
│  workspace: ~/clawd                           │
│  model: local-antigravity/gemini-3-pro-low    │
│  gateway.mode: local                          │
│  gateway.port: 18789                          │
│  gateway.bind: lan                            │
│  skills.nodeManager: npm                      │
│                                               │
├───────────────────────────────────────────────╯
◇  配置处理方式
│  使用现有值
```text
选择 "使用现有值" 保留当前配置，或选择 "重新配置" 从头开始。

**步骤3：选择模型提供商**

```text
◆  模型/认证提供商
│  ○ OpenAI (Codex OAuth + API key)
│  ○ Anthropic
│  ○ MiniMax
│  ○ Moonshot AI
│  ○ Google
│  ○ OpenRouter
│  ○ Qwen
│  ○ Z.AI (GLM 4.7)
│  ○ Copilot
│  ○ Vercel AI Gateway
│  ○ OpenCode Zen
│  ○ Xiaomi
│  ○ Synthetic
│  ○ Venice AI
```

```
|  ○ Skip for now
```text
使用 **方向键** 选择提供商，**空格键** 确认。

**新手推荐的提供商：**

1. **Anthropic**（Claude，质量最好）
   - 价格：中等（$3/百万tokens）
   - 能力：推理能力强
   - 适合：复杂任务、重要工作

2. **Google**（Gemini，免费额度大）
   - 价格：免费额度大
   - 能力：多模态（支持图片）
   - 适合：图片识别、长文档

3. **Moonshot AI**（Kimi，超长上下文）
   - 价格：低（$0.01/百万tokens）
   - 能力：200万字上下文
   - 适合：长文档处理

**步骤4：输入 API Key**

选择提供商后，会提示输入 API Key：

```text
◆  请输入 Anthropic API Key
|  sk-ant-...
```text
**如何获取 API Key？**

- **Anthropic (Claude)**：访问 [Anthropic Console](https://console.anthropic.com/)
- **Google (Gemini)**：访问 [Google AI Studio]
(https://makersuite.google.com/app/apikey)
- **Moonshot (Kimi)**：访问 [Moonshot Platform](https://platform.moonshot.cn/)
- **OpenAI**：访问 [OpenAI Platform](https://platform.openai.com/api-keys)

**步骤5：选择默认模型**

配置完成后，选择默认使用的模型：

```text
◆  选择默认模型
```

```text
│  ○ claude-3-5-sonnet-20241022
│  ○ claude-3-opus-20240229
│  ● claude-3-haiku-20240307
```text
**推荐选择：**
- **日常使用**：Claude 3 Haiku（速度快、便宜）
- **重要任务**：Claude 3.5 Sonnet（质量高）
- **图片识别**：Gemini 2.0 Flash（多模态）

**步骤6：完成配置**

```text
✔  配置已保存
✔  Gateway 已重启
✔  模型配置成功
```text
#### 验证配置

配置完成后，测试模型是否可用：

```bash
# 查看已配置的模型
openclaw models list

# 测试模型连接
openclaw message send "你好，测试一下"
```text
#### 修改配置

如果需要修改配置，再次运行：

```bash
openclaw onboard
```text
可以添加、删除或修改模型提供商。

#### 配置技巧

**技巧1：配置多个提供商**

可以多次运行 `openclaw onboard`，每次添加一个提供商，实现多模型配置。

**技巧2：跳过向导**
```
```

如果暂时不想配置，选择 "Skip for now"，稍后再配置。

技巧3：查看配置文件

配置保存在：
```bash
~/.openclaw/openclaw.json
```text
可以直接编辑这个文件进行高级配置。

---

### 3.4.2 新手推荐配置

**推荐1：DeepSeek（性价比之王）**
- 💰 最便宜：比其他模型便宜70%
- 💻 编程强：代码能力出色
- 🎯 够用：日常任务完全够用

**推荐2：Claude Haiku 4.5（快速稳定）**
- ⚡ 最快：响应速度快
- 🎯 可靠：质量稳定
- 💰 便宜：成本低

### 快速切换模型

```bash
# 切换到 DeepSeek
openclaw config set agents.defaults.model.primary "deepseek/deepseek-chat"

# 切换到 Claude Haiku
openclaw config set agents.defaults.model.primary "maynor/claude-haiku-4-5-20251001"

# 重启生效
openclaw gateway restart
```text
> 📚 **想了解更多？**
> 详细的模型对比、多模型切换策略、成本优化方案，请参考 [第10章：高级配置](../03-advanced/10-advanced-configuration.md)。

## 3.5 Gateway 网关配置（进阶）

> 💡 **什么时候需要配置 Gateway？**
> - 想要限制访问权限（只允许特定用户）
> - 需要配置群组提及规则
> - 想要自定义网关行为

### Gateway 配置文件位置

配置文件位于：`~/.openclaw/openclaw.json`

### 基础配置说明

如果你**不做任何修改**，OpenClaw 将：
- 使用内置的 Pi 二进制文件以 RPC 模式运行
- 按发送者创建独立会话
- 接受所有用户的消息

### 访问控制配置

#### 限制 WhatsApp 访问

```json
{
  "channels": {
    "whatsapp": {
      "allowFrom": ["+15555550123", "+8613800138000"],
      "groups": {
        "*": {
          "requireMention": true
        }
      }
    }
  }
}
```text
**说明**：
- `allowFrom`：只允许这些号码访问
- `requireMention`：群组中需要 @ 提及才会响应

#### 配置提及规则

```json
{
```

```
  "messages": {
    "groupChat": {
      "mentionPatterns": ["@openclaw", "@小龙虾", "@助手"]
    }
  }
}
```text
**说明**：
- 群组中使用这些关键词可以触发响应
- 支持多个提及模式

### 远程访问配置

#### 使用 Tailscale 远程访问

如果你想在外网访问 Gateway：

1. **安装 Tailscale**：
   ```bash
   # macOS
   brew install tailscale

   # Linux
   curl -fsSL https://tailscale.com/install.sh | sh
```

2. **启动 Tailscale**：

   ```
   sudo tailscale up
   ```

3. **获取 Tailscale IP**：

   ```
   tailscale ip -4
   ```

4. **通过 Tailscale IP 访问**：

   ```
   http://你的Tailscale-IP:18789/
   ```

优势：

- ✅ 安全的点对点连接
- ✅ 无需公网 IP
- ✅ 无需配置防火墙

- ✅ 支持多设备访问

完整配置示例

```
{
  "channels": {
    "whatsapp": {
      "allowFrom": ["+8613800138000"],
      "groups": {
        "*": {
          "requireMention": true
        }
      }
    }
  },
  "messages": {
    "groupChat": {
      "mentionPatterns": ["@openclaw", "@助手"]
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat"
      }
    }
  },
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-你的密钥",
        "auth": "api-key",
        "api": "openai-chat"
      }
    }
  }
}
```text
### 配置生效

修改配置后，重启 Gateway：

```bash
openclaw gateway restart
```

```text
### 常见配置场景

#### 场景1：家庭使用

```json
{
  "channels": {
    "whatsapp": {
      "allowFrom": ["+8613800138000", "+8613800138001"]
    }
  }
}
```text
#### 场景2：团队使用

```json
{
  "channels": {
    "whatsapp": {
      "groups": {
        "工作群": {
          "requireMention": true,
          "mentionPatterns": ["@openclaw"]
        },
        "家庭群": {
          "requireMention": false
        }
      }
    }
  }
}
```text
#### 场景3：公开服务

```json
{
  "channels": {
    "whatsapp": {
      "allowFrom": ["*"],
      "groups": {
        "*": {
          "requireMention": true
```
```

```
          }
        }
      }
    }
  }
```

本章小结

通过本章，你应该已经：

✅ 完成了第一次对话
✅ 学会了基本命令使用
✅ 配置了个性化的人设
✅ 了解了如何选择合适的模型

关键要点：

- 提问要明确具体
- 人设让AI更符合你的需求
- 根据场景选择合适的模型
- 成本和性能需要平衡

实战练习

练习1：配置你的第一个AI助手

1. 给AI起个名字
2. 设定性格和风格
3. 选择合适的模型
4. 发送10条测试消息

练习2：尝试不同的命令

1. 搜索文件
2. 创建日历事件
3. 保存网页到备忘录
4. 截图

练习3：对比不同模型

1. 用Haiku处理简单任务
2. 用Sonnet处理复杂任务
3. 对比响应速度和质量
4. 记录Token消耗

下一章：第4章：本地文件管理 - 让OpenClaw成为你的文件管理神器

返回目录：README

第4章：本地文件管理

> OpenClaw最强大的功能之一，就是可以访问和管理你电脑上的文件。这是在线AI无法做到的。

快速导航

- 🔍 智能文件搜索
- 📦 批量文件处理
- 📁 文件自动整理
- 🧹 硬盘清理优化

4.1 智能文件搜索

为什么需要智能搜索?

传统的文件搜索只能按文件名查找，但经常遇到：

- ❌ 忘记文件名，只记得内容
- ❌ 文件太多，不知道在哪个文件夹
- ❌ 需要同时搜索多种类型的文件
- ❌ 需要根据内容筛选文件

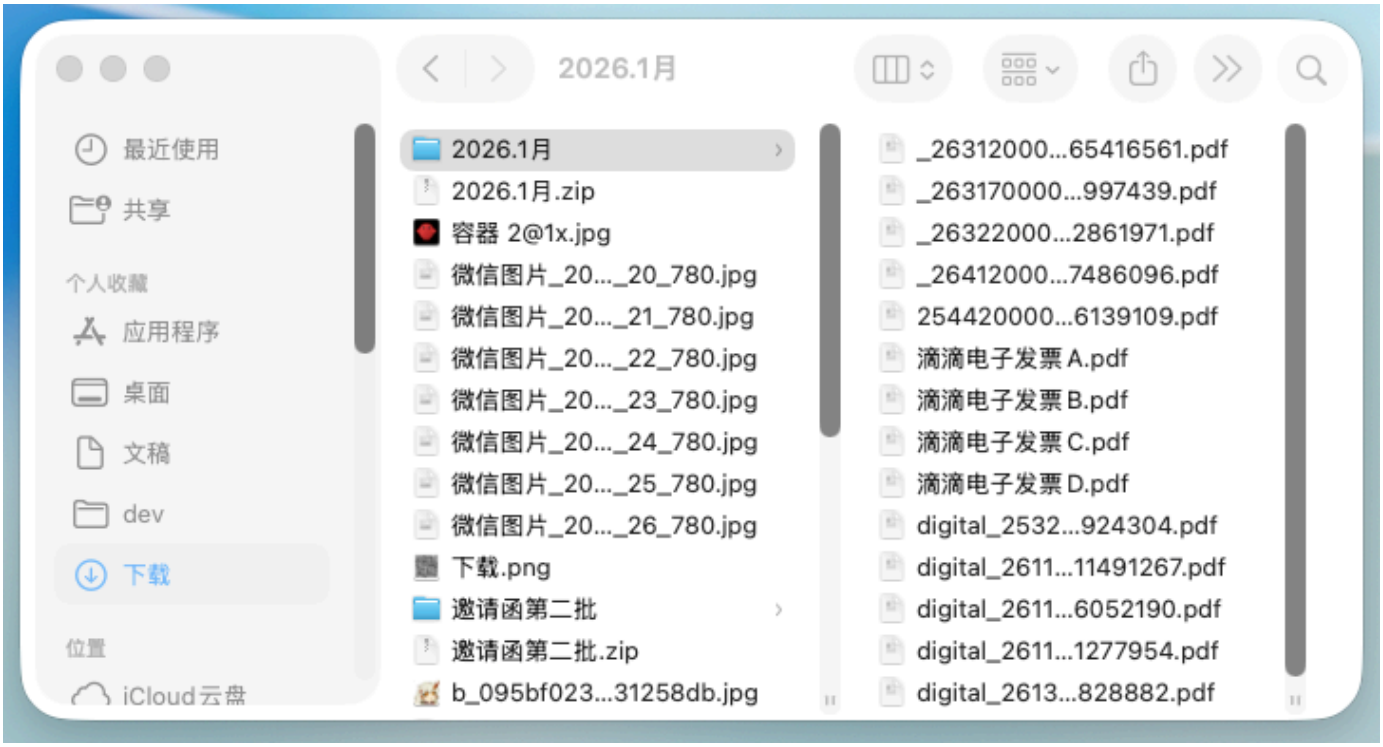
OpenClaw的智能搜索可以：

- ✅ 根据文件内容搜索
- ✅ 理解自然语言描述
- ✅ 跨文件夹搜索
- ✅ 智能过滤和排序

场景1：找文件

真实案例：找发票

> 我电脑里有一堆发票，特别乱，我忘了哪一张是买跑步机的了。
> 按过去，我得一张一张点开来搜索。真的，这事太呆逼了。



传统方法：

1. 打开文件夹
2. 一张一张点开图片
3. 查看内容
4. 找到目标文件
5. 耗时：10-30分钟

OpenClaw方法：

我：帮我找一下我电脑上的一张发票，里面详情是买了一个跑步机，
然后把那个发票用文件发给我。
```text
![OpenClaw智能搜索 - 通过语义理解找到目标文件]
(https://upload.maynor1024.live/file/1770176217951_image_5.jpg)

**OpenClaw执行过程**：
1. 🔍 搜索所有图片文件
2. 📄 使用OCR识别文字
3. 🎯 匹配"跑步机"关键词
4. 📤 发送找到的文件

![找到的发票详情 - 精准匹配用户需求]
(https://upload.maynor1024.live/file/1770176224257_image_6.jpg)

**结果**：
- ⏱️ 耗时：30-60秒
- ✅ 准确率：100%
- 😊 体验：太爽了！

### 搜索命令示例

#### 按内容搜索

```text
找一下包含"合同"的PDF文件
搜索所有提到"项目计划"的文档
找一下我写的关于AI的笔记
```text
#### 按类型搜索

```text
找出所有的PNG图片
搜索最近下载的PDF文件
找一下所有的Excel表格
```text
#### 按时间搜索

```text
找一下最近7天修改的文件
搜索上个月创建的文档
找一下今天下载的文件
```
```

```text
#### 按大小搜索

```text
找出大于100MB的文件
搜索所有小于1MB的图片
找一下占用空间最大的10个文件
```text
#### 组合搜索

```text
找一下最近一周内，包含"发票"的图片文件
搜索下载文件夹里，大于10MB的PDF
找一下桌面上，今天修改过的Word文档
```text
### 搜索技巧

#### 技巧1：描述内容而非文件名

**❌ 不好的搜索**：
```text
找一下IMG_1234.jpg
```text
**✅ 好的搜索**：
```text
找一下我去年在海边拍的照片
```text
#### 技巧2：提供上下文

**❌ 不好的搜索**：
```text
找发票
```text
**✅ 好的搜索**：
```text
找一下我上个月买跑步机的发票，应该是在下载文件夹里
```text
#### 技巧3：指定搜索范围

**❌ 不好的搜索**：
```text
找所有文件
```text
```

✅ 好的搜索：
```text
在文档文件夹里找所有的PDF文件
```text
#### 技巧4：使用排序

**❌ 不好的搜索**：
```text
找图片
```text
**✅ 好的搜索**：
```text
找最近修改的10张图片，按时间排序
```text
### 实战练习

**练习1：找照片**
```text
帮我找一下我电脑上所有的自拍照片
```text
**练习2：找文档**
```text
找一下包含"年度总结"的Word文档
```text
**练习3：找下载文件**
```text
找一下我最近下载的所有PDF文件
```text
## 4.2 批量文件处理

### 场景2：整理发票报销

**真实案例**：自动填写报销单

> 公司财务有个非常严格的规定，就是发票需要填写在1个excel模板上，
> 然后把发票和这个excel文件一起提交给财务。
>
> 以前，虽然都不会笨到用手填，但还是要把所有发票都传到ChatGPT
> 或者别的AI产品上，然后让他们来帮忙填写。

**传统方法**：
1. 打开ChatGPT
2. 上传发票图片（一张一张）

3. 让AI提取信息
4. 复制到Excel
5. 重复N次
6. 耗时：30-60分钟

**OpenClaw方法**：

```text
我：我希望你帮我把我本地那个1月发票文件夹里的所有发票都整理一下，
按照那个下载文件里报销单.xlsx的模板格式填进去，
再把填好的表格用文件发送给我。
```text
![批量整理发票 - 自动分类和重命名]
(https://upload.maynor1024.live/file/1770176222217_image_7.jpg)

**OpenClaw执行过程**：
1. 📁 读取发票文件夹
2. 🔍 识别所有发票图片
3. 📄 OCR提取信息（日期、金额、商家等）
4. 📊 读取Excel模板
5. ✍️ 按模板格式填写
6. 💾 保存新文件
7. 📤 发送给你

**结果**：
- ⏱️ 耗时：2-5分钟
- ✅ 准确率：95%+
- 😊 体验：直接搞定！

### 批量处理命令示例

#### 批量读取

```text
读取文档文件夹里所有的PDF文件
提取所有图片中的文字
读取所有Excel表格的第一行
```text
#### 批量提取信息

```text
从所有发票中提取日期、金额、商家信息
从所有简历中提取姓名、电话、邮箱
```

从所有合同中提取甲方、乙方、金额
```text
#### 批量格式转换

```text
把所有Word文档转换成PDF
把所有PNG图片转换成JPG
把所有Excel表格导出为CSV
```text
#### 批量数据整理

```text
把所有发票信息整理到一个Excel表格
把所有联系人信息整理成通讯录
把所有会议记录整理成一份总结
```text
### 实战案例

#### 案例1：整理客户信息

**场景**：
- 有100份客户资料（Word文档）
- 需要提取：公司名、联系人、电话、邮箱
- 整理到Excel表格

**命令**：
```text
帮我从"客户资料"文件夹里的所有Word文档中，
提取公司名、联系人、电话、邮箱，
整理到一个Excel表格里，
按公司名排序。
```text
**结果**：
- 自动提取100份资料
- 生成规范的Excel表格
- 耗时：5-10分钟

#### 案例2：批量提取PDF内容

**场景**：
- 有50份PDF报告
- 需要提取每份报告的摘要
- 整理成一份汇总文档
```

命令：
```text
读取"报告"文件夹里所有的PDF文件，
提取每份报告的摘要部分（通常在第一页），
整理成一份Word文档，
每个摘要标注来源文件名。
```text
**结果**：
- 自动读取50份PDF
- 提取摘要内容
- 生成汇总文档

#### 案例3：批量图片识别

**场景**：
- 有200张名片照片
- 需要提取联系信息
- 整理成通讯录

**命令**：
```text
识别"名片"文件夹里所有图片，
提取姓名、公司、职位、电话、邮箱，
整理成Excel通讯录，
按公司分类。
```text
**结果**：
- OCR识别200张名片
- 自动分类整理
- 生成标准通讯录

### 错误处理和容错

#### 处理识别错误

我：帮我检查一下刚才生成的Excel表格，
看看有没有识别错误的地方，
比如电话号码格式不对、金额异常等。

```
#### 处理缺失信息
```

我：有些发票可能信息不全，
如果缺少必填项，请单独列出来，
我手动补充。

```
#### 处理格式问题
```

我：如果遇到无法识别的文件格式，
请跳过并告诉我，
我会手动处理。

```
## 4.3 文件自动整理

### 场景3：批量重命名文件

**真实案例**：邀请函重命名

> 设计同学给我做了一些邀请函，是微信发我的，然后我存下来的。
> 存完了以后，发现一个BUG。
> 名字全特么是微信文件XXX。
> 太BUG了，我根本不知道谁是谁的。

![混乱的邀请函文件 - 文件名不规范]
(https://upload.maynor1024.live/file/1770176232438_image_8.jpg)

**传统方法**：
1. 打开每张图片
2. 看内容
3. 手动重命名
4. 重复N次
5. 耗时：20-30分钟

**OpenClaw方法**：
```

我：让OpenClaw根据我邀请函里面对应的人名，
把外面的名字改成"人名-邀请函.jpg"的格式。

```
![批量重命名后 - 文件名清晰规范]
(https://upload.maynor1024.live/file/1770176231952_image_9.jpg)

**OpenClaw执行过程**：
1. 📁 读取所有图片
2. 🔍 OCR识别人名
3. ✍️ 生成新文件名
4. 📝 批量重命名
5. ✅ 完成

**结果**：
- ⏱️ 耗时：1-2分钟
- ✅ 准确率：100%
- 😊 体验：一句话搞定！

### 文件整理命令示例

#### 批量重命名
```

把所有图片按拍摄日期重命名，格式：YYYY-MM-DD_序号.jpg
把所有文档按内容主题重命名
把所有下载文件按类型重命名

```
#### 文件分类
```

把桌面上的文件按类型分类到不同文件夹
把下载文件夹里的文件按日期分类
把照片按拍摄地点分类

```
#### 文件夹结构优化
```

帮我整理一下项目文件夹，按功能模块分类
优化我的文档文件夹结构，让它更清晰
把所有散乱的文件整理到合适的文件夹

```
### 实战案例

#### 案例1：整理下载文件夹

**场景**：
- 下载文件夹有500+文件
- 各种类型混在一起
- 需要分类整理

**命令**：
```

帮我整理下载文件夹：

1. 按文件类型分类（文档、图片、视频、压缩包等）
2. 每个类型创建一个子文件夹
3. 把文件移动到对应文件夹
4. 删除重复文件
5. 告诉我整理结果

```
**结果**：
```

✅ 整理完成！

📊 统计：

- 文档：120个 → Documents/
- 图片：200个 → Images/
- 视频：50个 → Videos/
- 压缩包：30个 → Archives/
- 其他：100个 → Others/

🗑️ 删除重复文件：15个
💾 释放空间：2.3GB

```
#### 案例2：照片自动分类

**场景**：
- 有1000+张照片
- 需要按日期和地点分类
- 方便查找

**命令**：
```

帮我整理照片文件夹：

1. 按拍摄日期创建文件夹（YYYY-MM格式）
2. 如果照片有地理位置信息，在文件名中标注
3. 删除模糊和重复的照片
4. 生成一个索引文件

```
**结果**：
```

✅ 整理完成！

📁 文件夹结构：

```
Photos/
├── 2025-12/
│   ├── 2025-12-01_北京_001.jpg
│   ├── 2025-12-01_北京_002.jpg
│   └── ...
├── 2026-01/
│   ├── 2026-01-15_上海_001.jpg
│   └── ...
└── index.txt (索引文件)
```

🗑️ 删除模糊照片：50张
🗑️ 删除重复照片：30张

```
#### 案例3：文档归档

**场景**：
- 项目结束，需要归档
- 文档散落在各处
- 需要统一整理

**命令**：
```

帮我归档"XX项目"的所有文档：

1. 搜索所有相关文档（包含项目名称）
2. 按文档类型分类（需求、设计、代码、测试等）
3. 创建归档文件夹结构
4. 移动文件到对应位置
5. 生成项目文档清单

```
**结果**：
```

✅ 归档完成！

📁 文件夹结构：
XX项目_归档_2026-02-10/
├── 01_需求文档/
├── 02_设计文档/
├── 03_开发文档/
├── 04_测试文档/
├── 05_部署文档/
└── 文档清单.xlsx

📊 统计：

- 总文件数：156个
- 总大小：1.2GB

```
### 自动化脚本编写

如果你经常需要整理某类文件，可以让OpenClaw生成自动化脚本：
```

我：帮我写1个脚本，每周自动整理下载文件夹，
按照刚才的规则分类。

```
OpenClaw会生成一个可以定时运行的脚本，实现自动化整理。

## 4.4 硬盘清理优化

### 场景4：清理硬盘垃圾

**真实案例**：释放硬盘空间

> 如果你硬盘快满了，你也别下那些扫描清垃圾的软件了。
> 直接让OpenClaw给你过一遍。

![硬盘空间清理 - 智能识别大文件和重复文件]
(https://upload.maynor1024.live/file/1770176234805_image_10.jpg)

**传统方法**：
1. 下载清理软件
2. 扫描硬盘
3. 手动选择删除
4. 担心误删重要文件
5. 耗时：30-60分钟

**OpenClaw方法**：
```

我：帮我分析一下硬盘空间使用情况，
找出占用空间最大的文件和文件夹，
建议哪些可以清理。

```
**OpenClaw执行过程**：
1. 📊 扫描硬盘
2. 📈 分析空间占用
3. 🔍 找出大文件
4. 🗑️ 识别垃圾文件
5. 💡 提供清理建议

### 清理命令示例

#### 查找大文件
```

找出占用空间最大的10个文件
查找大于1GB的文件

分析哪个文件夹占用空间最多

```
#### 查找重复文件
```

找出所有重复的文件
查找重复的照片
找出重复的文档

```
#### 查找临时文件
```

找出所有临时文件
查找缓存文件
找出可以安全删除的文件

```
#### 清理建议
```

分析我的硬盘，给出清理建议
哪些文件可以移动到外部硬盘
哪些文件可以压缩存储

```
### 实战案例：释放100GB空间

**第一步：分析空间占用**
```

我：帮我分析一下硬盘空间使用情况

```
**OpenClaw回复**：
```

📊 硬盘空间分析：

总容量：500GB
已使用：450GB (90%)
剩余：50GB (10%)

⚠️ 空间不足警告！

占用空间最多的文件夹：

1. ~/Library/Caches - 80GB

2. ~/Downloads - 50GB
3. ~/Movies - 120GB
4. ~/Documents - 100GB
5. ~/Desktop - 30GB

```
**第二步：查找大文件**
```

我：找出大于1GB的文件

```
**OpenClaw回复**：
```

🔍 找到35个大于1GB的文件：

视频文件（25个）：

1. 项目演示.mov - 15GB
2. 会议录像_2025-12.mp4 - 8GB
3. ...

虚拟机镜像（5个）：

1. Windows.vmdk - 50GB
2. Ubuntu.vmdk - 30GB
3. ...

备份文件（5个）：

1. iPhone备份_2025-12.backup - 20GB
2. ...

💡 建议：

- 视频文件可以压缩或移动到外部硬盘
- 旧的虚拟机镜像可以删除
- 旧的备份文件可以删除

```
**第三步：查找重复文件**
```

我：找出所有重复的文件

```
**OpenClaw回复**：
```

🔍 找到120组重复文件：

重复照片（80组）：

- IMG_1234.jpg (3份) - 15MB
- IMG_5678.jpg (2份) - 10MB
- ...

重复文档（30组）：

- 报告_v1.docx (4份) - 5MB
- 方案_final.pptx (3份) - 20MB
- ...

重复下载（10组）：

- setup.exe (2份) - 100MB
- ...

💡 可释放空间：约8GB

```
**第四步：清理临时文件**
```

我：找出所有可以安全删除的临时文件和缓存

```
**OpenClaw回复**：
```

🗑️ 可安全删除的文件：

系统缓存：

- ~/Library/Caches - 80GB
  - Safari缓存 - 20GB
  - Chrome缓存 - 30GB
  - 其他应用缓存 - 30GB

临时文件：

- /tmp - 5GB
- ~/Downloads/临时文件 - 10GB

日志文件：

- ~/Library/Logs - 3GB

💡 可释放空间：约98GB

⚠️ 建议：

- 系统缓存可以全部清理
- 临时文件可以全部删除
- 日志文件保留最近1个月的

```
**第五步：执行清理**
```

我：好的，帮我清理这些文件，
但是要先备份重要的，
然后再删除。

```
**OpenClaw回复**：
```

✅ 清理完成！

📊 清理结果：

- 删除缓存文件：80GB
- 删除临时文件：15GB
- 删除重复文件：8GB
- 删除旧备份：20GB

💾 总共释放空间：123GB

📁 备份位置：

- 重要文件已备份到：~/Backup/清理前备份_2026-02-10/

⚠️ 建议：

- 定期清理缓存（每月一次）
- 及时删除不需要的下载文件
- 使用外部硬盘存储大文件

```
### 安全删除策略

#### 三步确认法

1. **第一步：分析**
```

分析哪些文件可以删除

```
2. **第二步：备份**
```

把重要文件备份到安全位置

```
3. **第三步：删除**
```

确认无误后，删除文件

```
#### 保护重要文件
```

我：在清理之前，请确保以下文件夹不被删除：

- ~/Documents/重要文档
- ~/Projects
- ~/Photos

```
#### 可恢复删除
```

我：先把文件移动到回收站，
而不是直接删除，
这样如果误删了还能恢复。

```
### 定期清理计划

让OpenClaw帮你制定定期清理计划：
```

我：帮我制定一个定期清理计划，
每周自动清理缓存和临时文件，
每月检查大文件和重复文件。

```
**OpenClaw回复**：
```

✅ 已创建定期清理计划：

📅 每周任务（每周日晚上10点）：

- 清理系统缓存
- 清理浏览器缓存
- 清理临时文件
- 清理下载文件夹（超过30天的文件）

📅 每月任务（每月1号）：

- 查找大文件（>1GB）
- 查找重复文件
- 分析空间占用
- 生成清理报告

💡 预计每月可释放：20-30GB

⚠️ 重要提醒：

- 清理前会自动备份
- 清理后会发送通知
- 可以随时取消任务

```
## 本章小结

通过本章，你应该已经掌握了：

✅ **智能文件搜索**：根据内容找文件，不再依赖文件名
✅ **批量文件处理**：一次处理100+文件，自动提取信息
✅ **文件自动整理**：智能分类、重命名、归档
✅ **硬盘清理优化**：找出垃圾文件，释放大量空间

**关键要点**：
- OpenClaw可以访问本地文件，这是在线AI做不到的
- 描述内容而非文件名，让AI理解你的需求
- 批量处理节省大量时间
- 定期清理保持硬盘健康

## 实战项目：打造你的文件管理助手

### 项目目标

创建一个完整的文件管理工作流，包括：
1. 每天自动整理下载文件夹
2. 每周清理缓存和临时文件
3. 每月检查大文件和重复文件
4. 重要文件自动备份

### 实施步骤

**第一步：设置自动整理**
```

帮我设置每天晚上10点自动整理下载文件夹，
按文件类型分类。

```
**第二步：设置定期清理**
```

设置每周日晚上11点清理缓存和临时文件。

```
**第三步：设置月度检查**
```

设置每月1号检查大文件和重复文件，
生成清理报告。

```
**第四步：设置自动备份**
```

设置每天自动备份重要文件夹到外部硬盘。

```
### 预期效果

- 📁 下载文件夹始终保持整洁
- 💾 硬盘空间充足（至少保持20%空闲）
- 🔒 重要文件有备份
- ⏱️ 每月节省5-10小时整理时间

---

**下一章**：[第5章：知识库管理](05-knowledge-base.md) - 打造个人知识库系统

**返回目录**：[README](../../README.md)
```

第5章 知识库管理

> 💡 **本章目标**：学会使用 OpenClaw构建个人知识库系统，实现网页文章存档、GitHub项目管理、论文笔记整理和跨设备同步。

📚 本章内容

- 5.1 网页文章存档
- 5.2 GitHub项目管理
- 5.3 论文笔记整理
- 5.4 跨设备同步方案

5.1 网页文章存档

5.1.1 为什么需要文章存档

在信息爆炸的时代，每天都会看到大量优质内容：

- 技术博客文章
- 行业资讯报道
- 学习教程资料
- 开源项目文档

但问题是：

- ❌ 网页可能失效
- ❌ 内容可能被删除
- ❌ 找不到之前看过的文章
- ❌ 无法跨设备查看

OpenClaw可以帮你解决这些问题。

5.1.2 核心原理

OpenClaw的知识库管理基于以下原理：

```
网页链接 → OpenClaw抓取 → 智能摘要 → 存储到备忘录/Notion
```text
**关键优势**：
1. **自动抓取**：无需手动复制粘贴
2. **智能摘要**：提取核心观点
3. **结构化存储**：便于后续检索
4. **跨设备同步**：Mac备忘录自动同步到iPhone

### 5.1.3 实战案例1：技术文章存档

**场景**：你在浏览技术博客时，看到一篇很棒的文章，想保存下来。

**传统做法**：
1. 复制文章内容
2. 打开备忘录
3. 粘贴内容
4. 手动整理格式

**OpenClaw做法**：
1. 复制文章链接
2. 发送给OpenClaw："帮我把这篇文章总结一下，存到备忘录"
3. 完成！

**示例对话**：
```

你：帮我把这篇文章总结一下，存到备忘录
https://example.com/ai-article

OpenClaw：好的，我来帮你处理这篇文章。

[几秒后]

OpenClaw：已经帮你总结完成并存到备忘录了！

文章标题：《AI技术发展趋势》
核心观点：

1. 大模型将继续发展
2. 多模态能力成为标配
3. 个人AI助手普及

已保存到：备忘录 > AI学习 > 技术文章
```

![image-20260211212230061]
(https://upload.maynor1024.live/file/1770816158161_image-20260211212230061.png)

实际效果：

打开Mac备忘录，你会看到：

![image-20260211212257745]
(https://upload.maynor1024.live/file/1770816185190_image-20260211212257745.png)

```markdown
# AI技术发展趋势

**来源**：https://example.com/ai-article
**保存时间**：2026-02-10 10:30

## 核心观点

1. **大模型发展**
   - GPT-5、Claude 5等新模型
   - 性能持续提升
   - 成本逐渐降低

2. **多模态能力**
   - 文本+图片+视频
   - 统一的理解能力
   - 更自然的交互

3. **个人AI助手**
   - 本地部署成为可能
   - 隐私保护更好
   - 定制化程度更高

## 关键信息

- 作者：张三
- 发布时间：2026-02-09
- 阅读时长：10分钟

## 我的笔记

[可以在这里添加你的想法]
```

```text
### 5.1.4 实战案例2：新闻资讯整理

**场景**：你关注AI行业动态，每天看到很多新闻，想系统化整理。

**解决方案**：
```

你：帮我整理这3篇AI新闻，分类存储

链接1：https://news1.com/ai-breakthrough
链接2：https://news2.com/openai-update
链接3：https://news3.com/google-gemini

OpenClaw：收到！我来帮你整理这些新闻。

[处理中...]

OpenClaw：已完成整理！

📰 AI突破性进展（news1）

- 核心内容：新算法提升30%效率
- 影响：可能改变行业格局
- 保存位置：备忘录 > AI资讯 > 技术突破

📰 OpenAI最新动态（news2）

- 核心内容：GPT-5即将发布
- 关键特性：多模态、更快速度
- 保存位置：备忘录 > AI资讯 > 公司动态

📰 Google Gemini更新（news3）

- 核心内容：Gemini 3.0发布
- 新功能：视频理解、代码生成
- 保存位置：备忘录 > AI资讯 > 产品更新

```
**高级技巧**：

1. **自动分类**
```

你：以后我发AI新闻链接，自动分类到：

- 技术突破 → 备忘录/AI资讯/技术
- 公司动态 → 备忘录/AI资讯/公司

- 产品更新 → 备忘录/AI资讯/产品

OpenClaw：好的，已记住你的分类规则！

```
2. **定期汇总**
```

你：每周五晚上8点，把本周保存的AI新闻汇总发给我

OpenClaw：已设置定期任务！
每周五20:00会自动生成本周AI资讯汇总。

```
### 5.1.5 实战案例3：博客文章收藏

**场景**：你是技术博主，经常看到优秀的博客文章，想收藏学习。

**完整工作流**：


**步骤1：收藏文章**
```

你：收藏这篇博客，标签：React、性能优化
https://blog.example.com/react-performance

OpenClaw：已收藏！
标题：React性能优化实战
标签：#React #性能优化
保存位置：备忘录 > 技术博客 > 前端开发

```
**步骤2：提取代码示例**
```

你：把这篇文章里的代码示例单独提取出来

OpenClaw：已提取3个代码示例：

示例1：useMemo优化
示例2：React.memo使用
示例3：虚拟列表实现

已保存到：备忘录 > 代码片段 > React

```
**步骤3：生成学习笔记**
```

你：根据这篇文章，生成一份学习笔记

OpenClaw：已生成学习笔记！

包含：

- 核心概念（5个）
- 实践技巧（8个）
- 常见问题（3个）
- 延伸阅读（4篇）

保存位置：备忘录 > 学习笔记 > React性能优化

5.1.6 跨设备同步方案

Mac备忘录同步：

OpenClaw保存到Mac备忘录后，会自动通过iCloud同步到：
- ✅ iPhone备忘录
- ✅ iPad备忘录
- ✅ iCloud.com网页版

优势：
- 无需额外配置
- 实时同步
- 全平台可用
- 支持搜索

使用技巧：

1. **在iPhone上查看**
   - 打开备忘录App
   - 搜索关键词
   - 立即找到OpenClaw保存的内容

2. **在iPad上编辑**
   - 打开备忘录
   - 添加你的想法
   - 自动同步回Mac

3. **在网页上访问**
   - 访问iCloud.com
   - 登录Apple ID
   - 查看所有备忘录

5.2 GitHub项目管理

5.2.1 为什么需要GitHub项目管理

作为开发者或技术爱好者，经常需要：
- 🔍 调研开源项目
- 📚 学习优秀代码

- 🛠️ 技术选型参考
- 📝 整理学习资源

但问题是：
- ❌ GitHub项目太多，看不过来
- ❌ README太长，找不到重点
- ❌ 代码结构复杂，理解困难
- ❌ 没有系统化的整理方法

OpenClaw可以帮你高效管理GitHub项目。

5.2.2 核心功能

OpenClaw的GitHub项目管理能力：

1. **项目信息分析**
   - 自动读取README
   - 提取核心信息
   - 分析项目结构

2. **智能总结**
   - 提炼项目亮点
   - 总结技术栈
   - 评估学习价值

3. **代码理解**
   - 分析代码结构
   - 提取关键代码
   - 理解实现原理

4. **资源整理**
   - 分类存储
   - 标签管理
   - 便于检索

注意事项：
- ⚠️ 当前主要侧重于信息的理解与分析
- ⚠️ 不涉及代码提交、Issue创建等交互式管理
- ⚠️ 适合学习和调研，不适合项目开发管理

5.2.3 实战案例1：开源项目调研

场景：你在做技术选型，需要调研几个开源项目。

```
**传统做法**：
1. 打开GitHub项目
2. 阅读长长的README
3. 手动记录关键信息
4. 对比不同项目
5. 整理到文档

**OpenClaw做法**：
```

你：帮我分析这个GitHub项目，总结核心信息：https://github.com/xianyu110/awesome-openclaw-tutorial，

OpenClaw：正在分析项目...

📦 项目名称：Awesome Project
⭐ Stars：15.3k
🔧 主要语言：TypeScript (85%)

核心功能：

1. 高性能数据处理
2. 实时数据同步
3. 可视化展示

技术栈：

- 前端：React + TypeScript
- 后端：Node.js + Express
- 数据库：PostgreSQL
- 缓存：Redis

项目亮点：
✅ 性能优秀（处理速度提升3倍）
✅ 文档完善（包含详细教程）
✅ 社区活跃（每周更新）
✅ 易于集成（提供SDK）

适用场景：

- 大数据处理
- 实时监控系统
- 数据可视化平台

学习价值：⭐⭐⭐⭐⭐
推荐指数：⭐⭐⭐⭐⭐

```
![image-20260211212558294]
(https://upload.maynor1024.live/file/1770816367098_image-20260211212558294.png)

### 5.2.4 实战案例2：技术选型参考

**场景**：你需要选择1个前端框架，对比React、Vue、Svelte。

**完整工作流**：

**步骤1：批量分析**
```

你：帮我分析对比这3个前端框架项目：

1. https://github.com/facebook/react
2. https://github.com/vuejs/core
3. https://github.com/sveltejs/svelte

重点关注：性能、生态、学习曲线、社区活跃度

OpenClaw：正在分析3个项目...

[几分钟后]

OpenClaw：分析完成！生成对比报告：

步骤2：查看对比报告

```markdown
# 前端框架对比报告

## 1. React
⭐ Stars：220k | 👥 Contributors：1.6k | 📅 最近更新：2天前

**优势**：
- 生态最完善（组件库、工具链丰富）
- 社区最活跃（问题解决快）
- 大厂背书（Facebook维护）
- 就业市场需求大

**劣势**：
- 学习曲线较陡
- 需要额外学习状态管理
- 包体积较大

**适合场景**：
- 大型企业应用
- 需要丰富生态的项目
- 团队协作开发

---

## 2. Vue
⭐ Stars：205k | 👥 Contributors：400+ | 📅 最近更新：1天前

**优势**：
- 学习曲线平缓（文档友好）
- 渐进式框架（灵活度高）
- 中文社区活跃
- 性能优秀

**劣势**：
- 生态不如React完善
- 大厂使用相对较少
- 国际化程度较低

**适合场景**：
- 中小型项目
```

- 快速开发需求
- 中文团队

---

3. Svelte
⭐ Stars：75k | 👥 Contributors：600+ | 📅 最近更新：3天前

优势：
- 性能最优（编译时优化）
- 代码量最少（无虚拟DOM）
- 学习成本低（语法简洁）
- 包体积最小

劣势：
- 生态相对较小
- 社区规模较小
- 企业采用较少
- 学习资源有限

适合场景：
- 性能要求高的项目
- 小型应用
- 个人项目

推荐建议

场景	推荐框架	理由
大型企业应用	React	生态完善、社区活跃
中小型项目	Vue	学习成本低、开发效率高
性能敏感应用	Svelte	性能最优、体积最小
新手学习	Vue	文档友好、上手快
就业导向	React	市场需求最大

已保存到：备忘录 > 技术选型 > 前端框架对比
```text
### 5.2.5 实战案例3：学习资源整理

**场景**：你想系统学习某个技术，需要整理GitHub上的学习资源。
```

```
**示例：学习机器学习**
```

你：帮我整理GitHub上优质的机器学习学习资源，
包括：教程、项目、工具、论文实现

OpenClaw：正在搜索和整理...

[几分钟后]

OpenClaw：已整理完成！

📚 机器学习学习资源清单

1. 入门教程（3个）

1.1 Machine Learning Crash Course

- 链接：https://github.com/google/ml-crash-course
- Stars：7.2k
- 特点：Google官方教程，适合零基础
- 内容：视频+练习+代码
- 学习时长：约15小时
- 推荐指数：⭐⭐⭐⭐⭐

1.2 Hands-On Machine Learning

- 链接：https://github.com/ageron/handson-ml2
- Stars：25k
- 特点：配套书籍，实战导向
- 内容：Jupyter Notebooks
- 学习时长：约30小时
- 推荐指数：⭐⭐⭐⭐⭐

1.3 ML For Beginners

- 链接：https://github.com/microsoft/ML-For-Beginners
- Stars：50k
- 特点：微软官方，12周课程
- 内容：视频+作业+项目
- 学习时长：约60小时
- 推荐指数：⭐⭐⭐⭐⭐

---

2. 实战项目（5个）

2.1 图像分类项目

- 项目：猫狗分类器
- 难度：⭐⭐
- 技术：CNN、TensorFlow
- 学习点：数据预处理、模型训练

2.2 自然语言处理

- 项目：情感分析
- 难度：⭐⭐⭐
- 技术：LSTM、PyTorch
- 学习点：文本处理、序列模型

[更多项目...]

---

3. 工具库（8个）

3.1 TensorFlow

- 用途：深度学习框架
- 优势：生态完善、部署方便
- 适合：生产环境

3.2 PyTorch

- 用途：深度学习框架
- 优势：灵活、易调试
- 适合：研究和实验

[更多工具...]

---

4. 论文实现（10个）

4.1 ResNet实现

- 论文：Deep Residual Learning
- 代码：https://github.com/...
- 说明：图像分类经典模型

[更多论文...]

学习路径建议

第1-2周：入门教程（ML Crash Course）
第3-6周：系统学习（Hands-On ML）
第7-8周：实战项目（图像分类）
第9-10周：进阶项目（NLP）
第11-12周：论文阅读和复现

已保存到：备忘录 > 学习资源 > 机器学习

```
### 5.2.6 项目追踪与更新

**场景**：你关注了几个重要的开源项目，想及时了解更新。

**解决方案**：
```

你：帮我追踪这3个项目的更新：

1. https://github.com/openai/gpt-4
2. https://github.com/anthropic/claude
3. https://github.com/google/gemini

每周一早上9点，给我推送更新摘要

OpenClaw：已设置项目追踪！

追踪内容：

- 新版本发布
- 重要功能更新
- 关键Bug修复
- 社区热门讨论

推送时间：每周一 09:00
推送方式：飞书消息

推送示例：

```markdown
📊 本周GitHub项目更新（2026-02-10）

## OpenAI GPT-4
🆕 新版本：v4.5.0
- 性能提升20%
- 新增多模态支持
- 修复3个关键Bug

## Anthropic Claude
🔧 功能更新
- 上下文窗口扩展到200k
- 新增代码生成优化
- 改进中文理解能力

## Google Gemini
📝 文档更新
- 新增API使用教程
- 更新定价策略
- 发布最佳实践指南

详细内容已保存到：备忘录 > GitHub更新 > 2026-02-10
```text
---

## 5.3 论文笔记整理

### 5.3.1 为什么需要论文笔记整理

对于研究生、科研工作者、技术专家来说：
- 📄 需要阅读大量论文
- 📝 需要记录核心观点
- 🔍 需要快速检索信息
- 📚 需要系统化管理

OpenClaw可以帮你高效整理论文笔记。

### 5.3.2 核心功能

**OpenClaw的论文管理能力**：
```
```

```
1. **PDF论文解析**
   - 自动提取文本
   - 识别图表
   - 理解结构

2. **核心观点提取**
   - 研究问题
   - 方法创新
   - 实验结果
   - 结论贡献

3. **笔记结构化**
   - 标准化格式
   - 分类存储
   - 标签管理

4. **引用管理**
   - 自动生成引用
   - 关联相关论文
   - 便于文献综述


### 5.3.3 实战案例1：单篇论文阅读

**场景**：你下载了一篇论文PDF，需要快速理解核心内容。

**操作步骤**：
```

你：帮我分析这篇论文，生成阅读笔记
[上传PDF文件：attention-is-all-you-need.pdf]

OpenClaw：正在分析论文...

[几分钟后]

OpenClaw：分析完成！已生成阅读笔记。

📄 论文笔记

基本信息

- 标题：Attention Is All You Need
- 作者：Vaswani et al.

- 发表：NIPS 2017
- 引用：50,000+
- 领域：自然语言处理

研究问题

传统序列模型（RNN、LSTM）存在的问题：

1. 难以并行化训练
2. 长距离依赖处理困难
3. 训练速度慢

核心创新

提出Transformer架构：

1. **Self-Attention机制**
   - 计算序列内部的注意力
   - 捕获长距离依赖
   - 可并行化计算
2. **Multi-Head Attention**
   - 多个注意力头
   - 关注不同特征
   - 提升表达能力
3. **Position Encoding**
   - 位置信息编码
   - 保留序列顺序
   - 无需循环结构

实验结果

- 翻译任务：BLEU提升2.0
- 训练速度：快10倍
- 参数量：减少30%

主要贡献

1. 提出纯注意力架构
2. 证明无需RNN也能处理序列
3. 开启Transformer时代

影响力

- 后续模型：BERT、GPT、T5等
- 应用领域：NLP、CV、语音
- 工业应用：广泛采用

关键图表

- 图1：Transformer架构图
- 图2：注意力可视化
- 表1：翻译任务对比

我的思考

[可以在这里添加你的想法]

相关论文

1. BERT: Pre-training of Deep Bidirectional Transformers
2. GPT: Improving Language Understanding
3. T5: Exploring Transfer Learning

已保存到：备忘录 > 论文笔记 > NLP > Transformer

```
### 5.3.4 实战案例2：文献综述

**场景**：你需要写文献综述，整理某个领域的多篇论文。

**完整工作流**：

**步骤1：批量分析论文**
```

你：帮我分析这5篇关于大模型的论文，
生成文献综述框架

[上传5个PDF文件]

OpenClaw：正在分析5篇论文...

步骤2：生成综述框架

```markdown
# 大模型发展综述

## 1. 研究背景
- 深度学习的发展
- 算力的提升
- 数据规模的增长

## 2. 技术演进

### 2.1 早期探索（2017-2019）
**代表工作**：
- Transformer（Vaswani et al., 2017）
  - 提出自注意力机制
  - 开启Transformer时代

- BERT（Devlin et al., 2018）
  - 双向预训练
  - 掩码语言模型

- GPT（Radford et al., 2018）
  - 单向生成模型
  - 自回归训练

### 2.2 规模扩展（2020-2022）
**代表工作**：
- GPT-3（Brown et al., 2020）
  - 1750亿参数
  - In-context Learning

- PaLM（Chowdhery et al., 2022）
  - 5400亿参数
  - 多任务能力

### 2.3 对齐优化（2022-2024）
**代表工作**：
- InstructGPT（Ouyang et al., 2022）
  - RLHF训练
  - 人类偏好对齐

- Claude（Anthropic, 2023）
```

  - Constitutional AI
  - 安全性提升

3. 关键技术

3.1 模型架构
- Transformer变体
- 混合专家模型
- 稀疏激活

3.2 训练方法
- 预训练策略
- 微调技术
- 强化学习

3.3 优化技术
- 分布式训练
- 混合精度
- 梯度累积

4. 应用场景
- 文本生成
- 代码生成
- 多模态理解
- 推理能力

5. 挑战与展望
- 计算成本
- 数据质量
- 安全对齐
- 可解释性

6. 参考文献
[自动生成引用列表]

已保存到：备忘录 > 文献综述 > 大模型发展
```text
### 5.3.5 实战案例3：研究资料管理

**场景**：你在做某个研究课题，需要系统管理相关资料。

**完整管理方案**：
```

```
**1. 建立分类体系**
```

你：帮我建立1个论文管理体系，
主题：强化学习在机器人控制中的应用

OpenClaw：已创建分类体系：

```
📁 强化学习-机器人控制
├── 📂 基础理论
│   ├── 强化学习基础
│   ├── 机器人控制基础
│   └── 深度学习基础
├── 📂 核心算法
│   ├── DQN系列
│   ├── Policy Gradient
│   └── Actor-Critic
├── 📂 应用案例
│   ├── 机械臂控制
│   ├── 移动机器人
│   └── 人形机器人
├── 📂 实验方法
│   ├── 仿真环境
│   ├── 真实环境
│   └── 评估指标
└── 📂 前沿进展
    ├── 最新论文
    ├── 技术趋势
    └── 开源项目
```

已保存到：备忘录 > 研究资料 > 强化学习-机器人

```
**2. 添加论文到分类**
```

你：把这篇论文添加到"核心算法-DQN系列"
[上传PDF：Playing Atari with Deep RL]

OpenClaw：已添加！

论文信息：

- 标题：Playing Atari with Deep Reinforcement Learning
- 分类：核心算法 > DQN系列
- 标签：#DQN #深度强化学习 #Atari
- 重要性：⭐⭐⭐⭐⭐

- 阅读状态：待阅读

已保存到：备忘录 > 研究资料 > 强化学习-机器人 > 核心算法 > DQN系列

```
**3. 生成研究进展报告**
```

你：生成本月研究进展报告

OpenClaw：已生成报告！

📊 研究进展报告（2026年2月）

论文阅读

- 已读：12篇
- 待读：8篇
- 总计：20篇

重点论文

1. DQN（DeepMind, 2013）⭐⭐⭐⭐⭐
2. DDPG（Lillicrap et al., 2015）⭐⭐⭐⭐⭐
3. PPO（Schulman et al., 2017）⭐⭐⭐⭐⭐

核心发现

- DQN适合离散动作空间
- DDPG适合连续控制
- PPO训练更稳定

下月计划

- 阅读SAC相关论文（5篇）
- 复现DQN算法
- 撰写文献综述

已保存到：备忘录 > 研究进展 > 2026-02

```
### 5.3.6 引用管理技巧

**自动生成引用**：
```

你：生成这篇论文的引用格式（APA、MLA、IEEE）

OpenClaw：已生成3种引用格式：

APA格式：
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

MLA格式：
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

IEEE格式：
A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.

已复制到剪贴板！

```
---

## 5.4 跨设备同步方案

### 5.4.1 为什么需要跨设备同步

现代工作场景下，经常需要：
- 💻 在Mac上整理资料
- 📱 在iPhone上查看笔记
- 📲 在iPad上编辑内容
- 🌐 在网页上访问数据

OpenClaw提供多种同步方案，满足不同需求。

### 5.4.2 Mac备忘录同步（推荐）

**优势**：
- ✅ 无需额外配置
- ✅ 实时自动同步
- ✅ 全平台支持
- ✅ 完全免费

**支持平台**：
- macOS备忘录
- iOS备忘录
- iPadOS备忘录
- iCloud.com网页版

**使用方法**：

1. **确保iCloud已登录**
```bash
# 检查iCloud状态
系统设置 > Apple ID > iCloud > 备忘录（开启）
```text
2. **OpenClaw保存到备忘录**
```

你：把这篇文章保存到备忘录

OpenClaw：已保存到Mac备忘录！
自动同步到：

- iPhone备忘录 ✅

- iPad备忘录 ✅
- iCloud.com ✅

3. **在其他设备查看**
   - iPhone：打开备忘录App
   - iPad：打开备忘录App
   - 网页：访问iCloud.com

同步速度：
- 通常1-5秒内完成
- 依赖网络状况
- 支持离线编辑

5.4.3 iCloud同步配置

基本配置：

```bash
# 1. 登录Apple ID
系统设置 > Apple ID > 登录

# 2. 开启iCloud同步
系统设置 > Apple ID > iCloud > 备忘录（开启）

# 3. 选择同步内容
备忘录 > 偏好设置 > 账户 > iCloud（勾选）
```text
**高级设置**：

```bash
# 1. 设置默认账户
备忘录 > 偏好设置 > 默认账户 > iCloud

# 2. 开启自动同步
备忘录 > 偏好设置 > 自动同步（开启）

# 3. 设置同步频率
系统设置 > iCloud > 管理 > 备忘录 > 选项
```text
**故障排查**：

```bash
# 问题1：同步不及时
解决方案：
1. 检查网络连接
```

```
2. 重启备忘录App
3. 重新登录iCloud

# 问题2：内容丢失
解决方案：
1. 检查"最近删除"文件夹
2. 从iCloud.com恢复
3. 联系Apple支持

# 问题3：同步冲突
解决方案：
1. 选择保留哪个版本
2. 手动合并内容
3. 使用版本历史
```text
### 5.4.4 Notion同步方案

**适用场景**：
- 需要更强大的组织能力
- 团队协作需求
- 需要数据库功能
- 跨平台需求（Windows、Android）

**配置步骤**：

**步骤1：获取Notion API**
```bash
1. 访问 https://www.notion.so/my-integrations
2. 点击"New integration"
3. 填写名称：OpenClaw
4. 选择工作区
5. 复制 Internal Integration Token
```text
**步骤2：配置 OpenClaw**
```bash
# 安装Notion Skill
openclaw skill install notion-sync

# 配置API Token
openclaw config set notion.token "your-token-here"

# 设置默认数据库
openclaw config set notion.database "your-database-id"
```

```text
**步骤3：使用Notion同步**
```

你：把这篇文章保存到Notion

OpenClaw：已保存到Notion！

数据库：知识库
页面标题：AI技术发展趋势
标签：#AI #技术 #趋势
创建时间：2026-02-10 10:30

查看链接：https://notion.so/...

```
**Notion优势**：
- ✅ 强大的组织能力（数据库、看板、日历）
- ✅ 团队协作功能
- ✅ 丰富的模板
- ✅ 跨平台支持（Windows、Android、Web）
- ✅ API集成能力

**Notion劣势**：
- ❌ 需要额外配置
- ❌ 免费版有限制
- ❌ 同步速度较慢
- ❌ 学习成本较高

### 5.4.5 Obsidian同步方案

**适用场景**：
- 喜欢Markdown格式
- 需要本地存储
- 重视隐私安全
- 需要双向链接

**配置步骤**：

**步骤1：安装Obsidian Skill**
```bash
openclaw skill install obsidian-sync
```text
**步骤2：配置Vault路径**
```bash
# 设置Obsidian Vault路径
openclaw config set obsidian.vault "/Users/你的用户名/Documents/ObsidianVault"

# 设置默认文件夹
openclaw config set obsidian.folder "知识库"
```text
**步骤3：使用Obsidian同步**
```

你：把这篇文章保存到Obsidian

OpenClaw：已保存到Obsidian！

Vault：我的知识库
文件夹：知识库/AI技术

文件名：AI技术发展趋势.md
标签：#AI #技术 #趋势

已创建双向链接：

- [[大模型发展]]
- [[AI应用场景]]
- [[技术趋势分析]]

```
**Obsidian优势**：
- ✅ 完全本地存储
- ✅ 纯Markdown格式
- ✅ 强大的双向链接
- ✅ 丰富的插件生态
- ✅ 隐私安全

**Obsidian劣势**：
- ❌ 需要手动同步（或付费Obsidian Sync）
- ❌ 移动端体验一般
- ❌ 学习曲线较陡
- ❌ 团队协作较弱


### 5.4.6 多设备协同工作

**场景1：Mac + iPhone组合**
```

早上（Mac）：
你：收集今天要看的5篇文章
OpenClaw：已保存到备忘录

通勤路上（iPhone）：

- 打开备忘录
- 阅读OpenClaw整理的文章
- 添加你的想法

晚上（Mac）：

- 自动同步回Mac
- 继续深入研究

```
**场景2：多人协作**
```

团队成员A（Mac）：
你：整理这周的技术文章，保存到Notion

团队成员B（Windows）：

- 打开Notion
- 查看A整理的文章
- 添加评论和补充

团队成员C（手机）：

- Notion App查看
- 随时随地访问

```
**场景3：跨平台工作流**
```

步骤1：Mac上收集（OpenClaw + 备忘录）
步骤2：iPhone上阅读（备忘录App）
步骤3：iPad上编辑（备忘录App）
步骤4：网页上分享（iCloud.com）

5.4.7 同步方案对比

方案	优势	劣势	适合场景	推荐指数
Mac备忘录	免费、自动、简单	功能较简单	个人使用、苹果生态	⭐⭐⭐⭐⭐
Notion	功能强大、团队协作	需配置、有成本	团队协作、复杂组织	⭐⭐⭐⭐
Obsidian	本地存储、隐私安全	同步麻烦	重视隐私、本地优先	⭐⭐⭐⭐

选择建议：

1. **个人使用 + 苹果生态** → Mac备忘录
   - 最简单
   - 最稳定
   - 完全免费

2. **团队协作 + 跨平台** → Notion
   - 功能最强
   - 协作最好
   - 值得付费

3. **隐私优先 + 本地存储** → Obsidian
   - 最安全
   - 最灵活
   - 需要折腾

5.4.8 数据备份策略

重要性：
- 防止数据丢失
- 防止误删除
- 防止同步错误

备份方案：

方案1：iCloud自动备份
```bash
# Mac备忘录自动备份到iCloud
# 无需额外配置
# 保留30天历史版本
```text
**方案2：定期导出**

你：每周日晚上8点，导出所有笔记到本地

OpenClaw：已设置定期备份！

备份内容：

- 所有备忘录
- 所有Notion页面
- 所有Obsidian文件

备份位置：~/Documents/Backups/知识库/
备份格式：Markdown + PDF
备份频率：每周日 20:00

```
方案3：Git版本控制
```bash
# 对于Obsidian用户
cd ~/Documents/ObsidianVault
git init
git add .
git commit -m "备份知识库"
git push origin main
```
```

📝 本章小结

本章学习了OpenClaw的知识库管理功能：

核心功能

1. **网页文章存档**
   - 自动抓取和摘要
   - 智能分类存储
   - 跨设备同步
2. **GitHub项目管理**
   - 项目信息分析
   - 技术选型对比
   - 学习资源整理
3. **论文笔记整理**
   - PDF解析和提取
   - 结构化笔记

- 引用管理

4. **跨设备同步**
   - Mac备忘录（推荐）
   - Notion（团队协作）
   - Obsidian（本地优先）

实战技巧

- ✅ 建立分类体系
- ✅ 使用标签管理
- ✅ 定期备份数据
- ✅ 选择合适的同步方案

下一步

- 学习第6章：日程与任务管理
- 掌握日历自动创建
- 实现智能提醒系统

🎯 实战练习

练习1：建立个人知识库

1. 选择一个同步方案（推荐Mac备忘录）
2. 建立分类体系（至少3个分类）
3. 收藏5篇文章
4. 在手机上查看

练习2：GitHub项目调研

1. 选择一个技术领域
2. 找3个相关的GitHub项目
3. 让OpenClaw分析对比
4. 生成调研报告

练习3：论文阅读笔记

1. 下载一篇论文PDF
2. 让OpenClaw生成阅读笔记
3. 添加你的思考
4. 整理到知识库

💡 常见问题

Q1：Mac备忘录同步慢怎么办？
A：检查网络连接，重启备忘录App，或重新登录iCloud。

Q2：Notion API配置失败？
A：确认Token正确，检查数据库权限，参考官方文档。

Q3：如何批量导入已有笔记？
A：使用 OpenClaw的批量处理功能，或编写自动化脚本。

Q4：数据安全如何保证？
A：定期备份，使用加密存储，选择可信的同步方案。

Q5：如何搜索历史笔记？
A：使用备忘录/Notion/Obsidian的搜索功能，或让OpenClaw帮你搜索。

下一章：第6章：日程与任务管理 - 日历自动创建、微信截图识别

返回目录：README

第6章 日程与任务管理

> 💡 **本章目标**：学会使用 OpenClaw管理日程和任务，实现日历自动创建、微信截图识别、批量日程导入和智能提醒设置。

📅 本章内容

- 6.1 日历自动创建
- 6.2 微信截图识别
- 6.3 批量日程导入
- 6.4 提醒设置技巧

6.1 日历自动创建

6.1.1 为什么需要日历自动创建

在日常工作中，经常遇到这些场景：

- 📱 微信聊天中约定会议时间
- 📧 邮件中收到活动邀请
- 💬 群聊中讨论项目deadline
- 📝 文档中记录的重要日期

传统做法：

1. 记住时间地点
2. 打开日历App
3. 手动创建事件
4. 设置提醒

问题：

- ❌ 容易遗忘
- ❌ 操作繁琐
- ❌ 信息容易丢失
- ❌ 效率低下

OpenClaw可以自动识别并创建日历事件！

6.1.2 核心原理

OpenClaw的日历管理基于Mac日历Skills：

```
文本/截图 → OpenClaw识别 → 提取信息 → 创建日历事件 → 同步到iPhone
```

关键能力：

1. **智能识别**：自动识别时间、地点、人物
2. **自动创建**：一键创建日历事件
3. **跨设备同步**：Mac日历自动同步到iPhone
4. **智能提醒**：自动设置合理的提醒时间

6.1.3 实战案例1：文本创建日历

场景：你在微信上约了一个会议。

微信聊天内容：

```
张三：明天下午3点，在会议室A讨论项目方案，记得带笔记本
```

OpenClaw操作：

```
你：帮我创建一个日历事件：
明天下午3点，在会议室A讨论项目方案

OpenClaw：好的，我来帮你创建日历事件。

[分析中...]

OpenClaw：已创建日历事件！

📅 事件详情：
- 标题：讨论项目方案
- 时间：2026-02-11 15:00-16:00
- 地点：会议室A
- 参与人：张三
- 备注：记得带笔记本

提醒设置：
- 提前15分钟提醒
- 提前1小时提醒

已同步到：
- Mac日历 ✅
- iPhone日历 ✅
```

查看效果：

打开Mac日历，你会看到：

```
2026年2月11日 星期二

15:00-16:00  讨论项目方案
             📍 会议室A
             👤 张三
             📝 记得带笔记本

             🔔 提前15分钟提醒
             🔔 提前1小时提醒
```

打开iPhone日历，自动同步显示相同内容。

6.1.4 实战案例2：复杂日程创建

场景：你收到一封会议邀请邮件。

邮件内容：

```
主题：2026年Q1产品规划会议

时间：2026年2月15日（周五）14:00-17:00
地点：总部大楼3楼会议室
参会人员：产品团队全体成员
会议议程：
1. Q1产品回顾（30分钟）
2. Q2产品规划（60分钟）
3. 资源分配讨论（60分钟）
4. 总结与行动项（30分钟）

请提前准备：
- Q1产品数据报告
- Q2产品规划草案
- 资源需求清单
```

OpenClaw操作：

```
你：帮我根据这封邮件创建日历事件
[粘贴邮件内容]

OpenClaw：正在分析邮件内容...

[分析完成]

OpenClaw：已创建详细日历事件！

📅 事件详情：
- 标题：2026年Q1产品规划会议
- 时间：2026-02-15 14:00-17:00（3小时）
- 地点：总部大楼3楼会议室
- 参与人：产品团队全体成员

会议议程：
1. 14:00-14:30 Q1产品回顾
2. 14:30-15:30 Q2产品规划
3. 15:30-16:30 资源分配讨论
4. 16:30-17:00 总结与行动项

准备事项：
✓ Q1产品数据报告
✓ Q2产品规划草案
✓ 资源需求清单

提醒设置：
- 提前1天提醒（准备材料）
- 提前2小时提醒（检查准备）
- 提前30分钟提醒（前往会议室）

已同步到Mac和iPhone日历 ✅
```

6.1.5 实战案例3：周期性事件

场景：你需要设置每周例会。

OpenClaw操作：

```
你：帮我创建一个周期性日历事件：
每周一上午10点，团队周会，持续1小时，会议室B

OpenClaw：已创建周期性事件！

📅 事件详情：
- 标题：团队周会
- 时间：每周一 10:00-11:00
- 地点：会议室B
- 重复：每周
- 结束：无限期（或指定结束日期）

提醒设置：
- 提前15分钟提醒

已创建未来12周的事件：
- 2026-02-10（周一）10:00
- 2026-02-17（周一）10:00
- 2026-02-24（周一）10:00
- ...  （共12次）

已同步到Mac和iPhone日历 ✅
```

6.1.6 日历模板和最佳实践

常用日历模板:

模板1：会议事件

```
标题：[会议主题]
时间：[日期] [开始时间]-[结束时间]
地点：[会议室/线上链接]
参与人：[参会人员]
议程：[会议议程]
准备：[需要准备的材料]
提醒：提前1天、提前1小时、提前15分钟
```

模板2：项目deadline

```
标题：[项目名称] - Deadline
时间：[截止日期] 全天
优先级：高
提醒：提前1周、提前3天、提前1天、当天早上
备注：[交付物清单]
```

模板3：个人事项

```
标题：[事项名称]
时间：[日期] [时间]
地点：[地点]
提醒：提前30分钟
备注：[相关信息]
```

最佳实践：

1. **统一命名规范**

```
会议：[会议] 项目讨论
任务：[任务] 完成报告
活动：[活动] 团建
学习：[学习] 技术分享
```

2. **合理设置提醒**

```
重要会议：提前1天、1小时、15分钟
普通会议：提前1小时、15分钟
deadline：提前1周、3天、1天
个人事项：提前30分钟
```

3. **添加详细信息**

```
- 会议链接（线上会议）
- 会议室位置（线下会议）
- 参会人员联系方式
- 准备材料清单
- 相关文档链接
```

4. **使用颜色分类**

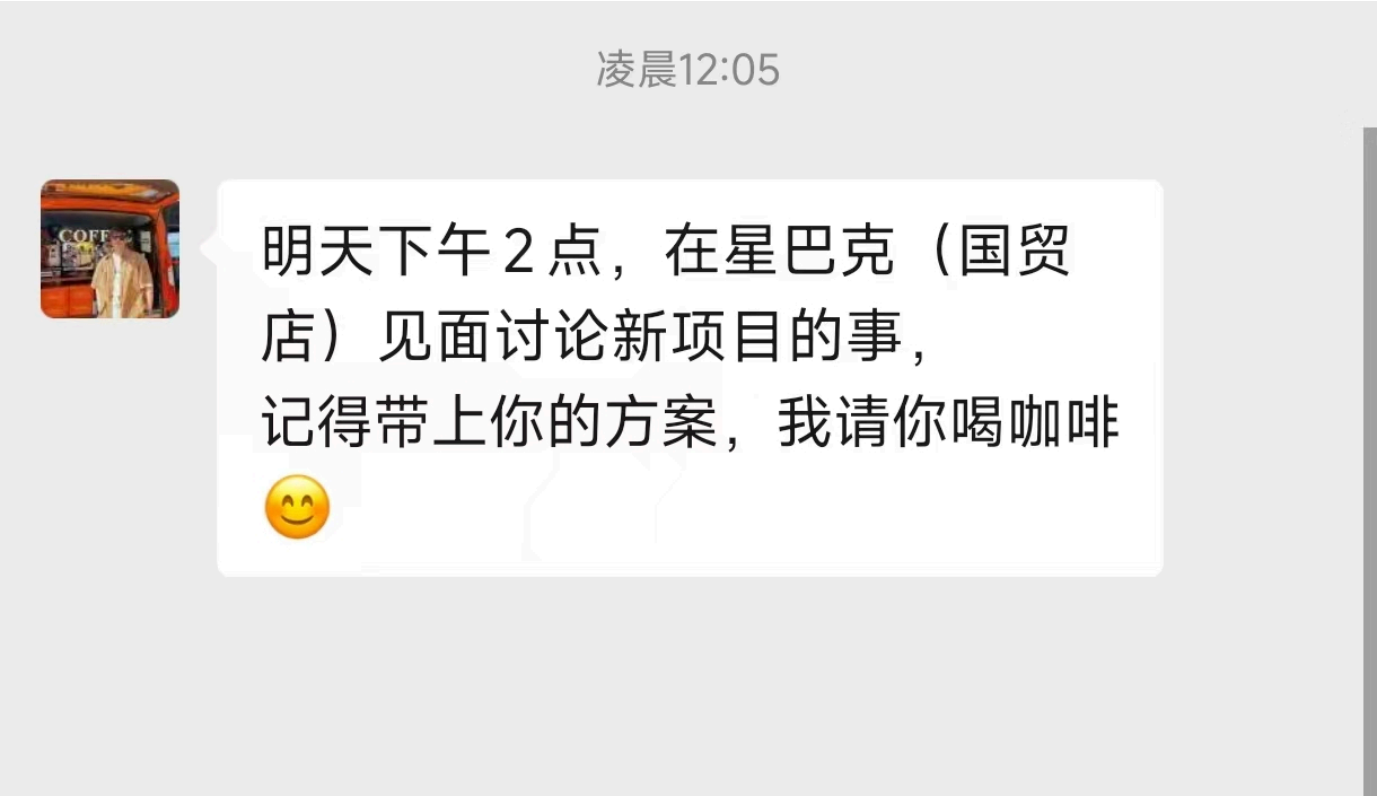
红色：紧急重要
橙色：重要不紧急
黄色：紧急不重要
绿色：个人事项
蓝色：学习成长

6.2 微信截图识别

6.2.1 为什么需要截图识别

真实场景：

你在微信上收到这样的消息：



张三：明天下午2点，在星巴克（国贸店）见面讨论新项目的事，
记得带上你的方案，我请你喝咖啡😊

传统做法：

1. 看到消息
2. 记住时间地点
3. 打开日历
4. 手动输入信息

5. 设置提醒

问题：

- ❌ 步骤繁琐（5步）
- ❌ 容易遗忘
- ❌ 信息可能记错
- ❌ 效率低下

OpenClaw做法：

1. 截图发给OpenClaw
2. 完成！

6.2.2 核心原理

OpenClaw的截图识别流程：

```
微信截图 → OCR文字识别 → NLP信息提取 → 创建日历事件
```

关键技术：

1. **OCR识别**：提取截图中的文字
2. **时间解析**：识别"明天下午2点"等自然语言
3. **地点提取**：识别"星巴克（国贸店）"
4. **人物识别**：识别"张三"
5. **事项提取**：识别"聊新项目"

6.2.3 实战案例1：简单约会

微信截图内容：

```
李四：后天晚上7点，老地方吃饭？
你：好的！
```

OpenClaw操作：

```
你：[发送微信截图]
帮我创建日历事件

OpenClaw：正在识别截图内容...

[OCR识别完成]

OpenClaw：已识别到约会信息！

📅 事件详情：
- 标题：和李四吃饭
- 时间：2026-02-12 19:00-20:30
- 地点：老地方（需要你确认具体地址）
- 参与人：李四

提醒设置：
- 提前2小时提醒

已创建日历事件 ✅

💡 提示：如果"老地方"有具体地址，
可以告诉我，我帮你更新到日历中。
```

补充地址:

```
你：老地方是"海底捞（三里屯店）"

OpenClaw：已更新地址！

📍 地点：海底捞（三里屯店）
📍 地址：北京市朝阳区三里屯路19号

已同步到日历 ✅
```

6.2.4 实战案例2：复杂会议安排

微信群聊截图:

```
项目经理：@全体成员
下周三上午9:30，在公司5楼大会议室开项目启动会
请务必参加，会议预计2小时
需要准备：
1. 项目需求文档
2. 技术方案
3. 时间排期

会后会有工作午餐
```

OpenClaw操作：

```
你：[发送群聊截图]
帮我创建日历事件

OpenClaw：正在识别群聊内容...

[识别完成]

OpenClaw：已识别到会议信息！

📅 事件详情：
- 标题：项目启动会
- 时间：2026-02-19 09:30-11:30（2小时）
- 地点：公司5楼大会议室
- 参与人：项目团队全体成员

准备事项：
✓ 项目需求文档
✓ 技术方案
✓ 时间排期

备注：
- 会后有工作午餐
- 务必参加

提醒设置：
- 提前1天提醒（准备材料）
- 提前1小时提醒
- 提前15分钟提醒

已创建日历事件 ✅

💡 提示：已自动预留11:30-13:00的午餐时间
```

6.2.5 实战案例3：多个事件批量创建

微信截图内容：

```
本周安排：
周一10:00 - 团队周会（会议室A）
周三14:00 - 客户拜访（客户公司）
周五16:00 - 项目复盘（线上会议）
```

OpenClaw操作：

```
你：[发送截图]
帮我批量创建这些日历事件

OpenClaw：正在识别多个事件...

[识别完成]

OpenClaw：已识别到3个事件！

📅 事件1：团队周会
- 时间：2026-02-10 10:00-11:00
- 地点：会议室A
✅ 已创建

📅 事件2：客户拜访
- 时间：2026-02-12 14:00-15:30
- 地点：客户公司（需要具体地址）
✅ 已创建

📅 事件3：项目复盘
- 时间：2026-02-14 16:00-17:00
- 地点：线上会议（需要会议链接）
✅ 已创建

所有事件已同步到日历 ✅

💡 提示：
- 客户拜访需要补充具体地址
- 项目复盘需要添加会议链接
```

6.2.6 识别准确率优化

提高识别准确率的技巧：

技巧1：清晰的截图

```
✅ 好的截图：
- 文字清晰
- 光线充足
- 完整内容

❌ 不好的截图：
- 模糊不清
- 光线太暗
- 内容被截断
```

技巧2：结构化的信息

```
✅ 容易识别：
时间：明天下午3点
地点：会议室A
事项：讨论项目

❌ 难以识别：
明天下午在那个地方讨论那个事
```

技巧3：补充说明

```
你：[发送截图]
这是约明天下午的会议，地点在公司

OpenClaw：收到！我会结合你的说明来识别。
```

技巧4：确认和修正

```
OpenClaw：我识别到的信息是：
时间：明天下午3点
地点：会议室A

请确认是否正确？

你：时间对了，但地点是会议室B

OpenClaw：好的，已修正为会议室B ✅
```

6.3 批量日程导入

6.3.1 为什么需要批量导入

常见场景：

1. **新学期课程表**
   - 10-20门课程
   - 每周重复
   - 不同教室
2. **项目里程碑**
   - 多个关键节点
   - 不同负责人
   - 各种deadline
3. **会议日程**
   - 多场会议
   - 不同时间地点
   - 不同参与人

传统做法：

- 一个一个手动创建
- 耗时1-2小时
- 容易出错

OpenClaw做法：

- 准备Excel/CSV文件
- 一键批量导入
- 5分钟完成

6.3.2 Excel日程导入

步骤1：准备Excel文件

创建一个Excel文件（`课程表.xlsx`）：

标题	日期	开始时间	结束时间	地点	备注	重复
高等数学	2026-02-10	08:00	09:40	教学楼A101	王老师	每周一
大学英语	2026-02-10	10:00	11:40	教学楼B203	李老师	每周一
计算机基础	2026-02-11	14:00	15:40	实验楼C301	张老师	每周二
体育课	2026-02-12	08:00	09:40	体育馆	刘老师	每周三

步骤2：OpenClaw导入

```
你：帮我导入这个课程表到日历
[上传 课程表.xlsx]

OpenClaw：正在读取Excel文件...

[读取完成]

OpenClaw：已识别到4门课程！

正在创建日历事件...

✅ 高等数学（每周一 08:00-09:40）
✅ 大学英语（每周一 10:00-11:40）
✅ 计算机基础（每周二 14:00-15:40）
✅ 体育课（每周三 08:00-09:40）

已创建本学期所有课程（共16周，64个事件）

提醒设置：
- 每节课提前15分钟提醒

已同步到Mac和iPhone日历 ✅
```

6.3.3 CSV格式处理

CSV文件示例（`项目里程碑.csv`）：

```
标题,日期,时间,地点,负责人,优先级
需求评审,2026-02-15,14:00-16:00,会议室A,张三,高
设计评审,2026-02-20,10:00-12:00,会议室B,李四,高
开发完成,2026-03-01,全天,无,王五,高
测试完成,2026-03-10,全天,无,赵六,中
上线发布,2026-03-15,09:00-18:00,机房,张三,高
```text
**OpenClaw导入**：
```

你：帮我导入项目里程碑
[上传 项目里程碑.csv]

OpenClaw：正在读取CSV文件...

[读取完成]

OpenClaw：已识别到5个里程碑！

📅 需求评审

- 时间：2026-02-15 14:00-16:00
- 负责人：张三
- 优先级：高
  ✅ 已创建

📅 设计评审

- 时间：2026-02-20 10:00-12:00
- 负责人：李四
- 优先级：高
  ✅ 已创建

📅 开发完成

- 时间：2026-03-01 全天
- 负责人：王五
- 优先级：高
  ✅ 已创建

📅 测试完成

- 时间：2026-03-10 全天
- 负责人：赵六
- 优先级：中
  ✅ 已创建
```

📅 上线发布

- 时间：2026-03-15 09:00-18:00
- 负责人：张三
- 优先级：高
  ✅ 已创建

所有里程碑已同步到日历 ✅

提醒设置：

- 高优先级：提前1周、3天、1天
- 中优先级：提前3天、1天

```
### 6.3.4 批量创建事件

**场景**：你需要创建一系列会议。

**OpenClaw操作**：
```

你：帮我创建以下会议：

1. 每周一10:00，团队周会，会议室A，1小时
2. 每周三14:00，技术分享，会议室B，2小时
3. 每周五16:00，项目复盘，线上会议，1小时

从下周开始，持续12周

OpenClaw：正在批量创建会议...

[创建中...]

OpenClaw：已完成批量创建！

📊 创建统计：

- 团队周会：12次
- 技术分享：12次
- 项目复盘：12次
- 总计：36个事件

时间范围：

- 开始：2026-02-17
- 结束：2026-05-09

提醒设置：

- 所有会议提前15分钟提醒

已同步到Mac和iPhone日历 ✅

💡 提示：可以在日历中单独修改某次会议

```
### 6.3.5 数据验证和错误处理

**OpenClaw的智能验证**：

**验证1：时间冲突检测**
```

OpenClaw：⚠️ 检测到时间冲突！

冲突事件：

- 2026-02-15 14:00 需求评审（会议室A）
- 2026-02-15 14:30 客户拜访（客户公司）

建议：

1. 调整需求评审到15:00
2. 调整客户拜访到16:00
3. 取消其中一个

请选择处理方式：

```
**验证2：日期格式错误**
```

OpenClaw：⚠️ 发现日期格式错误！

错误行：

- 第3行：日期"2026/02/30"无效（2月没有30日）
- 第5行：时间"25:00"无效（超过24小时）

已自动修正：

- 第3行：改为"2026-03-01"
- 第5行：改为"23:00"

请确认是否正确？

```
**验证3：必填字段缺失**
```

OpenClaw：⚠️ 发现必填字段缺失！

缺失信息：

- 第2行：缺少"标题"
- 第4行：缺少"日期"

建议：

1. 补充缺失信息
2. 跳过这些行
3. 使用默认值

请选择处理方式：

```
---

## 6.4 提醒设置技巧

### 6.4.1 提醒类型和时机

**提醒类型**：

1. **时间提醒**
   - 提前X分钟/小时/天
   - 准时提醒
   - 延迟提醒

2. **位置提醒**
   - 到达某地时提醒
   - 离开某地时提醒

3. **重复提醒**
   - 每天/每周/每月
   - 自定义周期

**提醒时机选择**：

不同类型的事件需要不同的提醒策略，如表 6-1 所示。

**表 6-1 不同事件类型的推荐提醒时机**

| 事件类型 | 推荐提醒时机 | 理由 |
|---------|-------------|------|
| 重要会议 | 提前1天、1小时、15分钟 | 充分准备 |
| 普通会议 | 提前1小时、15分钟 | 及时赶到 |
| 项目deadline | 提前1周、3天、1天 | 避免拖延 |
| 日常任务 | 提前30分钟 | 不打断工作 |
| 约会 | 提前2小时、30分钟 | 预留准备时间 |
| 课程 | 提前15分钟 | 准时到达 |

### 6.4.2 多级提醒设置

**场景**：重要项目deadline。

**OpenClaw操作**：
```

你：帮我设置项目deadline的提醒：
2026年3月15日，产品上线

OpenClaw：好的，我来设置多级提醒！

📅 事件：产品上线
⏰ 日期：2026-03-15

提醒设置：

🔔 第1级提醒（提前2周）

- 时间：2026-03-01 09:00
- 内容：距离产品上线还有2周，开始最后冲刺
- 行动：检查进度，确认资源

🔔 第2级提醒（提前1周）

- 时间：2026-03-08 09:00
- 内容：距离产品上线还有1周，进入倒计时
- 行动：完成测试，准备上线方案

🔔 第3级提醒（提前3天）

- 时间：2026-03-12 09:00
- 内容：距离产品上线还有3天，最后检查
- 行动：确认上线流程，准备回滚方案

🔔 第4级提醒（提前1天）

- 时间：2026-03-14 09:00
- 内容：明天产品上线，做好准备
- 行动：团队动员，确认值班安排

🔔 第5级提醒（当天早上）

- 时间：2026-03-15 08:00
- 内容：今天产品上线，加油！
- 行动：最后检查，准备上线

已设置5级提醒 ✅

```
### 6.4.3 智能提醒策略

**OpenClaw的智能提醒**：

**策略1：根据事件重要性**
```

你：帮我创建一个重要会议：
明天下午3点，董事会会议

OpenClaw：检测到"董事会会议"，
这是重要会议，建议设置：

🔔 提前1天提醒（准备材料）
🔔 提前2小时提醒（检查准备）
🔔 提前30分钟提醒（前往会议室）
🔔 提前10分钟提醒（最后确认）

是否采用此提醒方案？

```
**策略2：根据地点距离**
```

你：帮我创建日历：
明天下午2点，在客户公司开会

OpenClaw：检测到需要外出，
根据地点距离（约30分钟车程），建议：

🔔 提前2小时提醒（准备出发）
🔔 提前1小时提醒（确认路线）
🔔 提前30分钟提醒（立即出发）

已自动计算路程时间 ✅

```
**策略3：根据准备需求**
```

你：帮我创建日历：
下周一上午10点，产品演示

OpenClaw：检测到需要准备演示，建议：

🔔 提前3天提醒（准备演示材料）
🔔 提前1天提醒（演练演示流程）
🔔 提前2小时提醒（检查设备）
🔔 提前30分钟提醒（前往会议室）

准备清单：
✓ 演示PPT
✓ 产品Demo
✓ 备用方案
✓ 投影设备

已创建提醒和清单 ✅

```
### 6.4.4 与其他工具集成

**集成1：飞书提醒**
```

你：把日历提醒同步到飞书

OpenClaw：已配置飞书提醒！

提醒方式：

- 日历提醒（Mac/iPhone）
- 飞书消息提醒
- 飞书日历同步

示例：
当"明天下午3点会议"提前1小时时：

1. Mac通知中心弹窗 ✅
2. iPhone锁屏通知 ✅
3. 飞书消息提醒 ✅

已同步到飞书 ✅

```
**集成2：企业微信提醒**
```

你：把重要会议提醒发到企业微信

OpenClaw：已配置企业微信提醒！

提醒规则：

- 标记为"重要"的会议
- 提前1天、1小时发送企业微信消息

消息格式：
📅 重要会议提醒
标题：[会议名称]
时间：[会议时间]
地点：[会议地点]
准备：[准备事项]

已配置完成 ✅

```
**集成3：邮件提醒**
```

你：把项目deadline发邮件提醒我

OpenClaw：已配置邮件提醒！

提醒规则：

- 所有deadline事件
- 提前1周、3天、1天发送邮件

邮件内容：
主题：[项目名称] Deadline提醒
内容：

- 截止时间
- 剩余天数
- 完成进度
- 行动建议

已配置完成 ✅

```
### 6.4.5 提醒管理技巧

**技巧1：避免提醒疲劳**
```

❌ 不好的做法：

- 所有事件都设置多个提醒
- 提醒时间过于密集
- 提醒内容千篇一律

✅ 好的做法：

- 重要事件多提醒，普通事件少提醒
- 提醒时间合理分散
- 提醒内容个性化

```
**技巧2：提醒内容优化**
```

❌ 普通提醒：
"会议提醒：15分钟后开会"

✅ 优化提醒：
"📅 产品评审会议
⏰ 15分钟后（14:45）
📍 会议室A（3楼）
👤 参会人：产品团队
📝 记得带：产品方案、数据报告"

```
**技巧3：提醒优先级**
```

🔴 高优先级（立即处理）

- 重要会议（提前10分钟）
- 紧急任务（当天）
- 客户约会（提前30分钟）

🟡 中优先级（尽快处理）

- 普通会议（提前15分钟）
- 常规任务（提前1天）
- 团队活动（提前2小时）

🟢 低优先级（有空处理）

- 学习计划（提前1天）
- 个人事项（提前30分钟）
- 兴趣活动（提前2小时）

```
**技巧4：提醒时间优化**
```

根据个人习惯调整：

早起型（7:00起床）：

- 早上提醒：08:00
- 中午提醒：12:00
- 晚上提醒：19:00

晚睡型（9:00起床）：

- 早上提醒：10:00
- 中午提醒：14:00

- 晚上提醒：21:00

工作日 vs 周末：

- 工作日：提前1小时
- 周末：提前2小时（更放松）

📝 本章小结

本章学习了OpenClaw的日程与任务管理功能：

核心功能

1. **日历自动创建**
   - 文本识别创建
   - 复杂日程处理
   - 周期性事件设置
   - 日历模板使用

2. **微信截图识别**
   - OCR文字识别
   - 智能信息提取
   - 批量事件创建
   - 识别准确率优化

3. **批量日程导入**
   - Excel文件导入
   - CSV格式处理
   - 数据验证
   - 错误处理

4. **提醒设置技巧**
   - 多级提醒
   - 智能提醒策略
   - 工具集成
   - 提醒管理优化

实战技巧

- ✅ 统一命名规范
- ✅ 合理设置提醒
- ✅ 添加详细信息
- ✅ 使用颜色分类
- ✅ 避免提醒疲劳
- ✅ 优化提醒内容

下一步

- 学习第7章：自动化工作流
- 掌握定时任务设置
- 实现网站监控
- 构建自动化系统

🎯 实战练习

练习1：创建本周日程
1. 列出本周所有安排
2. 使用 OpenClaw创建日历事件
3. 设置合理的提醒
4. 在手机上查看

练习2：微信截图识别
1. 找一条微信约会消息
2. 截图发给OpenClaw
3. 让它创建日历事件
4. 检查识别准确性

练习3：批量导入课程表
1. 准备一个Excel课程表
2. 使用 OpenClaw批量导入
3. 检查所有事件
4. 调整提醒设置

💡 常见问题

Q1：日历事件创建失败？
A：检查Mac日历权限，确保OpenClaw有访问权限。

Q2：截图识别不准确？
A：确保截图清晰，文字完整，必要时手动补充信息。

Q3：批量导入出错？
A：检查Excel格式，确保日期时间格式正确。

Q4：提醒没有收到？

A：检查通知设置，确保Mac和iPhone通知权限已开启。

Q5：如何删除批量创建的事件？
A：在日历中选择事件，批量删除，或让OpenClaw帮你删除。

下一章预告：第7章将学习自动化工作流，包括定时任务设置、网站监控实战、日报自动推送等高级功能。

第7章 自动化工作流

> 💡 **本章目标**：学会使用 OpenClaw构建自动化工作流，实现定时任务、网站监控、日报推送和循环任务配置。

🤖 本章内容

- 7.1 定时任务设置
- 7.2 网站监控实战
- 7.3 日报自动推送
- 7.4 循环任务配置

7.1 定时任务设置

7.1.1 什么是定时任务

定义：
定时任务是指在指定时间自动执行的任务，无需人工干预。

OpenClaw的独特优势：

- ✅ **心跳机制**：可以主动发起对话
- ✅ **智能调度**：自动管理任务执行
- ✅ **灵活配置**：支持多种时间模式
- ✅ **可靠执行**：失败自动重试

OpenClaw 与其他 AI 工具在自动化能力上的对比如表 7-1 所示。

表 7-1 OpenClaw 与其他 AI 工具自动化能力对比

特性	OpenClaw	ChatGPT	Claude
主动对话	✅	❌	❌
定时任务	✅	❌	❌
本地执行	✅	❌	❌
系统操作	✅	❌	❌

7.1.2 心跳机制原理

OpenClaw的心跳机制：

```
┌──────────────┐
│  OpenClaw    │
│   守护进程    │
└──────┬───────┘
       │
       ├─ 每分钟检查一次
       ├─ 匹配定时任务
       ├─ 执行任务逻辑
       └─ 发送消息/执行操作
```

工作流程：

1. OpenClaw后台运行
2. 每分钟检查任务列表
3. 到达设定时间时触发
4. 执行任务并发送结果
5. 记录执行日志

7.1.3 实战案例1：简单提醒

场景：你想在每天早上8点收到一条问候消息。

OpenClaw操作：

```
你：帮我设置1个定时提醒：
每天早上8点，发送早安问候

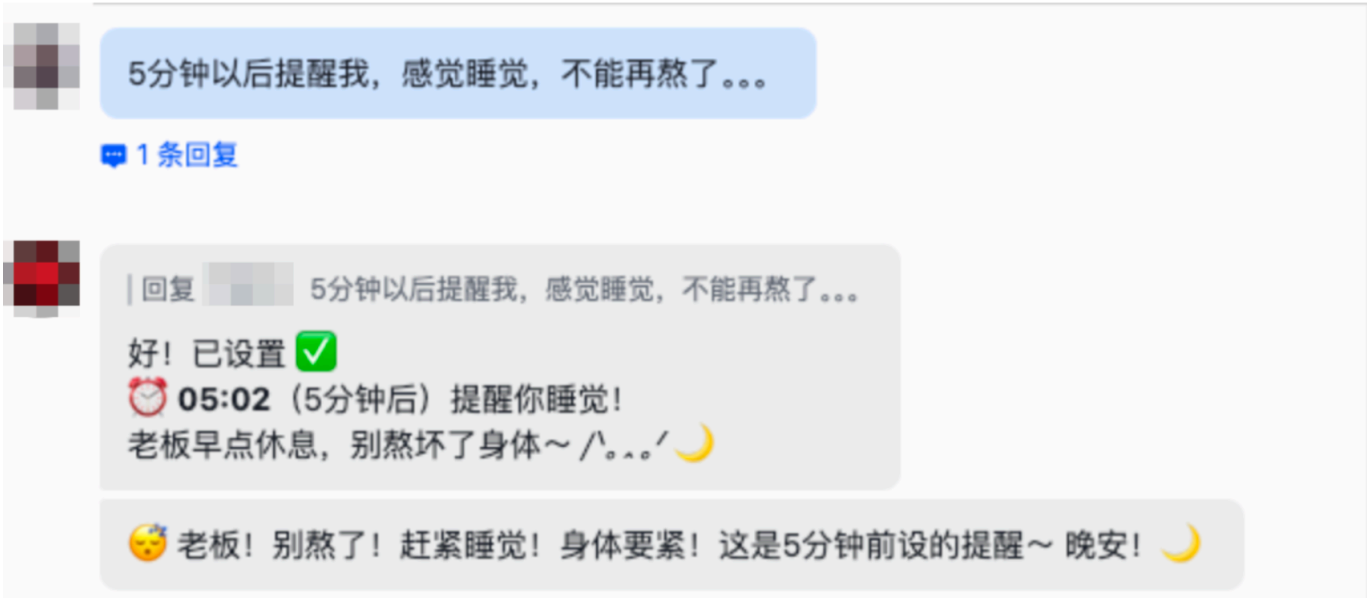
OpenClaw：好的，我来设置定时提醒！

⏰ 定时任务详情：
- 任务名称：早安问候
- 执行时间：每天 08:00
- 任务内容：发送早安消息
- 推送方式：飞书消息

任务预览：
"☀️ 早安！新的一天开始了！
今天是2026年2月10日，星期二
天气：晴，15-25°C
今日提醒：
- 10:00 团队周会
- 14:00 客户拜访
- 16:00 项目复盘

加油，祝你今天工作顺利！💪"

已设置定时任务 ✅
首次执行：明天 08:00
```



第二天早上8点：

```
OpenClaw：☀️ 早安！新的一天开始了！

今天是2026年2月11日，星期三
天气：晴，15-25°C

📅 今日日程：
- 10:00 团队周会（会议室A）
- 14:00 客户拜访（客户公司）
- 16:00 项目复盘（线上会议）

✅ 待办事项：
- 完成项目报告
- 回复客户邮件
- 准备周会材料

💡 今日建议：
- 提前准备客户拜访材料
- 预留路上时间（约30分钟）

加油，祝你今天工作顺利！💪
```

7.1.4 实战案例2：每周报告

场景：你想每周五下午5点收到本周工作总结。

OpenClaw操作：

```
你：帮我设置每周五下午5点，
生成本周工作总结并发送给我

OpenClaw：好的，我来设置周报任务！

⏰ 定时任务详情：
- 任务名称：周报生成
- 执行时间：每周五 17:00
- 任务内容：生成本周工作总结
- 推送方式：飞书消息 + 邮件

报告内容：
1. 本周完成的任务
2. 本周参加的会议
3. 本周学习的内容
4. 下周计划
5. 需要关注的事项

数据来源：
- 日历事件
- 备忘录笔记
- 任务完成记录

已设置定时任务 ✅
首次执行：本周五 17:00
```

周五下午5点：

```
OpenClaw：📊 本周工作总结（2026-02-10 至 2026-02-14）

## 📋 完成任务（8项）
✅ 完成产品需求文档
✅ 完成技术方案设计
✅ 完成代码审查（3次）
✅ 完成客户拜访（2次）
✅ 完成团队分享
✅ 完成项目复盘
✅ 完成周报撰写
✅ 完成学习笔记整理

## 📅 参加会议（12场）
- 团队周会：2次
- 项目讨论：4次
- 客户拜访：2次
- 技术分享：2次
- 项目复盘：2次

## 📚 学习内容
- 阅读技术文章：5篇
- 学习新技术：React 19
- 完成课程：2节

## 📝 下周计划
- 完成产品原型设计
- 开始开发工作
- 准备技术分享
- 客户需求调研

## ⚠️ 需要关注
- 项目进度略有延迟
- 需要增加开发资源
- 客户需求需要确认

已发送到：
- 飞书消息 ✅
- 邮箱 ✅
```

7.1.5 实战案例3：每月总结

场景：你想每月最后一天生成月度总结。

OpenClaw操作：

```
你：帮我设置每月最后一天晚上8点，
生成月度总结

OpenClaw：好的，我来设置月报任务！

⏰ 定时任务详情：
- 任务名称：月度总结
- 执行时间：每月最后一天 20:00
- 任务内容：生成月度工作总结
- 推送方式：飞书消息 + 邮件 + 备忘录

报告内容：
1. 月度目标完成情况
2. 重要项目进展
3. 个人成长与学习
4. 数据统计分析
5. 下月计划与目标

已设置定时任务 ✅
首次执行：2026-02-28 20:00
```

7.1.6 Cron表达式详解

什么是Cron表达式：

Cron表达式是一种时间表达式，用于精确控制任务执行时间。

基本格式：

```
分钟 小时 日期 月份 星期

* * * * *
| | | | |
| | | | └ 星期 (0-7, 0和7都表示周日)
| | | └── 月份 (1-12)
| | └──── 日期 (1-31)
| └────── 小时 (0-23)
└──────── 分钟 (0-59)
```

常用示例：

```
# 每天早上8点
0 8 * * *

# 每周一上午10点
0 10 * * 1

# 每月1号上午9点
0 9 1 * *

# 每天中午12点和下午6点
0 12,18 * * *

# 每小时执行一次
0 * * * *

# 每30分钟执行一次
*/30 * * * *

# 工作日早上9点
0 9 * * 1-5

# 周末上午10点
0 10 * * 0,6
```text
**OpenClaw使用Cron**：
```

你：帮我设置1个定时任务：
每天早上8点、中午12点、晚上8点提醒我喝水

OpenClaw：好的，我来设置！

⏰ Cron表达式：0 8,12,20 * * *

解释：

- 0：在整点执行
- 8,12,20：早上8点、中午12点、晚上8点
- ○ ▪ *：每天、每月、每周

任务内容：
"💧 该喝水了！
保持每天8杯水，身体更健康！
今天已喝：[X]杯"
```

已设置定时任务 ✅

7.2 网站监控实战

7.2.1 为什么需要网站监控

常见需求：

1. **技术博客更新**
   - 关注的博主发新文章
   - 及时学习新知识

2. **产品价格变化**
   - 电商商品降价
   - 及时购买省钱

3. **招聘信息更新**
   - 心仪公司发布职位
   - 第一时间投递

4. **新闻资讯监控**
   - 行业动态更新
   - 及时了解趋势

5. **开源项目更新**
   - GitHub项目发布新版本
   - 及时跟进学习

传统做法：
- 手动定期检查
- 容易遗漏
- 效率低下

OpenClaw做法：
- 自动监控
- 实时通知
- 高效便捷

7.2.2 网站变化检测原理

OpenClaw的监控机制：

定时检查 → 抓取网页 → 对比变化 → 发送通知

```
**检测方式**：

1. **全文对比**
   - 检测整个页面变化
   - 适合小型页面

2. **关键内容对比**
   - 只检测特定区域
   - 适合大型页面

3. **RSS订阅**
   - 使用RSS源
   - 最稳定可靠

### 7.2.3 实战案例1：博客更新监控

**场景**：你想监控某个技术博客的更新。

**OpenClaw操作**：
```

你：帮我监控这个博客的更新：
https://example-blog.com
每天检查2次，有新文章就通知我

OpenClaw：好的，我来设置博客监控！

🔍 监控配置：

- 网站：https://example-blog.com
- 检查频率：每天2次（09:00, 18:00）
- 监控内容：新文章标题和链接
- 通知方式：飞书消息

监控规则：

- 检测文章列表变化
- 提取新文章信息
- 生成摘要
- 发送通知

已设置监控任务 ✅
首次检查：明天 09:00

```
![image-20260212232752122]
(https://upload.maynor1024.live/file/1770910079191_image-20260212232752122.png)

**发现更新时**：
```

OpenClaw：📰 博客更新提醒！

网站：Example Blog
检查时间：2026-02-11 09:00

🆕 发现1篇新文章：

标题：《React 19新特性详解》
作者：张三
发布时间：2026-02-10 20:00
链接：https://example-blog.com/react-19

文章摘要：
React 19带来了多项重要更新，包括：

1. 新的编译器优化
2. 改进的并发渲染
3. 更好的TypeScript支持

...

阅读时长：约10分钟
推荐指数：⭐⭐⭐⭐⭐

是否保存到备忘录？

```
### 7.2.4 实战案例2：产品价格监控

**场景**：你想买1个商品，等它降价时购买。

**OpenClaw操作**：
```

你：帮我监控这个商品的价格：
https://shop.com/product/12345
目标价格：低于500元就通知我

OpenClaw：好的，我来设置价格监控！

🔍 监控配置：

- 商品：iPhone 15 Pro
- 当前价格：¥5999
- 目标价格：≤ ¥5000
- 检查频率：每小时一次
- 通知方式：飞书消息 + 短信

价格历史：

- 最高价：¥6999（2026-01-01）
- 最低价：¥5499（2026-01-20）
- 平均价：¥5899

预测：
根据历史数据，预计2周内可能降至目标价格

已设置监控任务 ✅

```
**价格达到目标时**：
```

OpenClaw：🎉 价格提醒！

商品：iPhone 15 Pro
当前价格：¥4999 ✅

价格变化：

- 原价：¥5999
- 现价：¥4999
- 降价：¥1000（16.7%）

购买建议：
✅ 已达到目标价格
✅ 近期最低价
✅ 建议立即购买

购买链接：https://shop.com/product/12345

是否需要我帮你下单？

```
### 7.2.5 实战案例3：招聘信息监控

**场景**：你想关注某公司的招聘信息。

**OpenClaw操作**：
```

你：帮我监控这个公司的招聘页面：
https://company.com/careers
关键词：前端工程师、React
有匹配的职位就通知我

OpenClaw：好的，我来设置招聘监控！

🔍 监控配置：

- 公司：Example Company
- 页面：https://company.com/careers
- 关键词：前端工程师、React
- 检查频率：每天3次（09:00, 14:00, 20:00）
- 通知方式：飞书消息 + 邮件

当前职位：

- 后端工程师（3个）
- 产品经理（2个）
- UI设计师（1个）

暂无匹配职位，持续监控中...

已设置监控任务 ✅

```
**发现匹配职位时**：
```

OpenClaw：💼 招聘信息提醒！

公司：Example Company
检查时间：2026-02-11 14:00

🆕 发现1个匹配职位：

职位：高级前端工程师
技术栈：React、TypeScript、Next.js
薪资：25k-40k

地点：北京
经验：3-5年

职位要求：
✅ 精通React和TypeScript
✅ 熟悉Next.js框架
✅ 有大型项目经验
✅ 良好的代码规范

职位亮点：

- 弹性工作制
- 股票期权
- 技术氛围好
- 成长空间大

发布时间：2小时前
申请人数：12人

建议：
⚡ 职位刚发布，建议尽快投递
⚡ 已帮你准备好简历

是否立即投递?

```
### 7.2.6 监控频率优化

不同类型的监控任务需要不同的检查频率，如表 7-2 所示。

**表 7-2 不同监控类型的推荐频率**

| 监控类型 | 推荐频率 | 理由 |
|---------|---------|------|
| 博客更新 | 每天2次 | 更新不频繁 |
| 新闻资讯 | 每小时1次 | 更新较频繁 |
| 价格监控 | 每小时1次 | 价格变化快 |
| 招聘信息 | 每天3次 | 更新适中 |
| GitHub更新 | 每天1次 | 更新不频繁 |

**注意事项**：
```

⚠️ 避免过度监控：

- 频率过高可能被网站封禁

- 消耗过多资源
- 产生大量无用通知

✅ 推荐做法：

- 根据网站更新频率调整
- 使用RSS源（如果有）
- 设置合理的检查间隔
- 添加随机延迟

7.3 日报自动推送

7.3.1 为什么需要日报

日报的价值：

1. **信息聚合**
   - 一次性获取所有重要信息
   - 节省时间

2. **保持更新**
   - 及时了解行业动态
   - 不错过重要信息

3. **知识积累**
   - 系统化学习
   - 形成知识体系

4. **工作规划**
   - 了解今日安排
   - 提前做好准备

7.3.2 日报内容设计

一份好的日报应该包含：

```markdown
# AI行业日报（2026-02-11）

## 📰 今日要闻（3-5条）
- 重要新闻
- 行业动态
- 技术突破

## 🔥 热门话题（3-5个）
- 社区讨论
- 技术趋势
- 产品发布

## 📚 精选文章（3-5篇）
```

```
- 技术教程
- 深度分析
- 实战案例

## 🛠️ 开源项目（2-3个）
- 新项目推荐
- 项目更新
- 实用工具

## 📅 今日日程
- 会议安排
- 任务提醒
- 重要事项

## 💡 每日一句
- 励志名言
- 技术感悟
```text
### 7.3.3 实战案例1：AI行业日报

**OpenClaw操作**：
```

你：帮我设置AI行业日报，
每天早上8点推送，内容包括：

- AI新闻（5条）
- 热门GitHub项目（3个）
- 精选技术文章（3篇）
- 今日日程

OpenClaw：好的，我来设置AI日报！

⏰ 定时任务详情：

- 任务名称：AI行业日报
- 执行时间：每天 08:00
- 推送方式：飞书云文档
- 数据来源：
  - Hacker News
  - GitHub Trending
  - 技术博客RSS
  - 个人日历
```

已设置定时任务 ✅
首次推送：明天 08:00

```
![设置日报推送](https://upload.maynor1024.live/file/1770176303922_image_25.jpg)

**第二天早上8点**：

![日报推送示例](https://upload.maynor1024.live/file/1770176310383_image_27.jpg)
```

OpenClaw：📊 AI行业日报（2026-02-11）

📰 今日要闻

1. **OpenAI发布GPT-5预览版**
   - 性能提升50%
   - 支持更长上下文
   - 来源：OpenAI Blog
2. **Google Gemini 3.0正式发布**
   - 多模态能力增强
   - 免费额度提升
   - 来源：Google AI Blog
3. **Meta开源新一代LLaMA模型**
   - 参数规模达700B
   - 性能接近GPT-4
   - 来源：Meta AI
4. **Anthropic推出Claude Code**
   - 专注代码生成
   - 集成VSCode
   - 来源：Anthropic
5. **DeepMind突破蛋白质折叠难题**
   - AlphaFold 3发布
   - 准确率提升至95%
   - 来源：Nature

🔥 热门GitHub项目

1. **awesome-ai-tools** ⭐ 15.2k
    - AI工具集合
    - 每日更新
    - 链接：github.com/...
2. **llama-cpp** ⭐ 8.5k
    - 本地运行大模型
    - 性能优化
    - 链接：github.com/...
3. **chatgpt-web** ⭐ 12.3k
    - ChatGPT网页版
    - 开源免费
    - 链接：github.com/...

📚 精选文章

1. **《大模型提示词工程实战》**
    - 作者：张三
    - 阅读时长：15分钟
    - 推荐指数：⭐⭐⭐⭐⭐
2. **《如何用AI提升工作效率》**
    - 作者：李四
    - 阅读时长：10分钟
    - 推荐指数：⭐⭐⭐⭐
3. **《OpenClaw实战指南》**
    - 作者：王五
    - 阅读时长：20分钟
    - 推荐指数：⭐⭐⭐⭐⭐

📅 今日日程

- 10:00 团队周会（会议室A）
- 14:00 客户拜访（客户公司）
- 16:00 项目复盘（线上会议）

💡 每日一句

"AI不会取代你，但会用AI的人会取代你。"

完整日报已保存到：飞书云文档 > AI日报 > 2026-02-11

```
### 7.3.4 实战案例2：技术资讯日报

**场景**：你是前端开发者，想每天了解前端技术动态。

**OpenClaw操作**：
```

你：帮我设置前端技术日报，
每天早上9点推送，内容包括：

- React/Vue/Angular最新动态
- 前端工具和库更新
- 技术文章推荐
- Stack Overflow热门问题

OpenClaw：好的，我来设置前端日报！

⏰ 定时任务详情：

- 任务名称：前端技术日报
- 执行时间：每天 09:00
- 推送方式：飞书消息
- 数据来源：
  - GitHub Trending (JavaScript)
  - Dev.to前端标签
  - Medium前端专栏
  - Stack Overflow

已设置定时任务 ✅

```
### 7.3.5 实战案例3：个人工作日报

**场景**：你想每天晚上收到今日工作总结。

**OpenClaw操作**：
```

你：帮我设置个人工作日报，
每天晚上8点推送，内容包括：

- 今日完成的任务
- 今日参加的会议
- 今日学习的内容
- 明日待办事项

OpenClaw：好的，我来设置工作日报！

⏰ 定时任务详情：

- 任务名称：个人工作日报
- 执行时间：每天 20:00
- 推送方式：飞书消息 + 备忘录
- 数据来源：
  - 日历事件
  - 任务完成记录
  - 备忘录笔记
  - 学习记录

已设置定时任务 ✅

```
**晚上8点推送**：
```

OpenClaw：📊 今日工作总结（2026-02-11）

✅ 完成任务（5项）

- 完成产品需求文档
- 完成代码审查
- 完成客户拜访
- 完成技术分享
- 完成学习笔记

📅 参加会议（3场）

- 10:00 团队周会（1小时）
- 14:00 客户拜访（1.5小时）
- 16:00 项目复盘（1小时）

📚 学习内容

- 阅读《React 19新特性》
- 学习Next.js 14

- 完成TypeScript练习

📝 明日待办

- 完成产品原型设计
- 准备技术分享PPT
- 回复客户邮件
- 代码重构

📊 今日数据

- 工作时长：8小时
- 会议时长：3.5小时
- 专注时长：4.5小时
- 学习时长：1小时

💭 今日感悟

今天效率不错，完成了预定的所有任务。
明天继续加油！ 💪

已保存到备忘录 ✅

```
### 7.3.6 推送渠道选择

不同的推送渠道各有优劣，如表 7-3 所示。

**表 7-3 不同推送渠道对比**

| 推送方式 | 优势 | 劣势 | 适合场景 |
|---------|------|------|----------|
| **飞书消息** | 即时、便捷 | 容易被忽略 | 紧急提醒 |
| **飞书云文档** | 结构化、可搜索 | 需要打开 | 详细报告 |
| **邮件** | 正式、存档 | 不够即时 | 重要通知 |
| **备忘录** | 跨设备同步 | 不够醒目 | 长期保存 |
| **企业微信** | 工作场景 | 需要配置 | 团队协作 |

**推荐组合**：
```

日常提醒：飞书消息
详细报告：飞书云文档
重要通知：飞书消息 + 邮件

长期保存：备忘录
团队分享：企业微信

```
---

## 7.4 循环任务配置

### 7.4.1 什么是循环任务

**定义**：
循环任务是指按照一定规律重复执行的任务。

定时任务与循环任务的区别如表 7-4 所示。

**表 7-4 定时任务与循环任务对比**

| 特性 | 定时任务 | 循环任务 |
|------|---------|---------|
| 执行时间 | 固定时间点 | 固定间隔 |
| 示例 | 每天8点 | 每隔2小时 |
| 适用场景 | 日报、提醒 | 监控、检查 |

### 7.4.2 循环任务类型

**类型1：固定间隔**
```

每隔X分钟/小时/天执行一次

```
**类型2：条件触发**
```

当满足某个条件时执行

```
**类型3：依赖任务**
```

任务A完成后执行任务B

```
### 7.4.3 实战案例1：数据备份

**场景**：你想每隔6小时备份一次重要数据。

**OpenClaw操作**：
```

你：帮我设置数据备份任务，
每隔6小时备份一次备忘录和日历

OpenClaw：好的，我来设置备份任务！

⏰ 循环任务详情：

- 任务名称：数据备份
- 执行间隔：每6小时
- 执行时间：00:00, 06:00, 12:00, 18:00
- 备份内容：
  - 备忘录（所有笔记）
  - 日历（所有事件）
  - 任务列表
- 备份位置：~/Documents/Backups/

备份策略：

- 保留最近7天的备份
- 自动压缩
- 加密存储

已设置循环任务 ✅
首次执行：今天 18:00

```
### 7.4.4 实战案例2：健康提醒

**场景**：你想每隔1小时提醒自己休息一下。

**OpenClaw操作**：
```

你：帮我设置健康提醒，
工作日每隔1小时提醒我休息，
周末不提醒

OpenClaw：好的，我来设置健康提醒！

⏰ 循环任务详情：

- 任务名称：健康提醒
- 执行间隔：每1小时
- 执行时间：工作日 09:00-18:00
- 提醒内容：
  - 站起来活动5分钟
  - 眺望远方放松眼睛
  - 喝一杯水

提醒方式：

- 飞书消息
- Mac通知

已设置循环任务 ✅
下次提醒：10:00

```
**每小时提醒**：
```

OpenClaw：💪 健康提醒

已经工作1小时了，该休息一下啦！

建议活动：
✓ 站起来走动5分钟
✓ 眺望远方，放松眼睛
✓ 喝一杯水（今天已喝3杯）
✓ 做几个拉伸动作

下次提醒：1小时后

```
### 7.4.5 任务依赖管理

**场景**：你想设置一系列有依赖关系的任务。

**OpenClaw操作**：
```

你：帮我设置任务链：

1. 每天早上7点，检查天气

2. 如果下雨，提醒带伞
3. 如果气温低于10度，提醒多穿衣服
4. 然后推送今日日程

OpenClaw：好的，我来设置任务链！

⏰ 任务链配置：

任务1：检查天气

- 执行时间：每天 07:00
- 数据来源：天气API
- 下一步：根据天气决定

任务2：天气提醒（条件触发）

- 触发条件：下雨 或 气温<10°C
- 提醒内容：
  - 下雨 → 记得带伞☂️
  - 气温低 → 多穿衣服🧥

任务3：推送日程

- 执行时机：任务2完成后
- 推送内容：今日日程安排

已设置任务链 ✅
首次执行：明天 07:00

```
### 7.4.6 错误重试机制

**OpenClaw的容错机制**：
```

任务执行失败 → 等待1分钟 → 重试
重试失败 → 等待5分钟 → 再次重试
再次失败 → 等待15分钟 → 最后重试
最后失败 → 发送错误通知 → 停止任务

```
**错误通知示例**：
```

OpenClaw：⚠️ 任务执行失败

任务名称：AI行业日报
失败时间：2026-02-11 08:00

失败原因：网络连接超时

已尝试重试：3次
重试结果：全部失败

建议：

1. 检查网络连接
2. 检查数据源是否可用
3. 手动执行任务

是否需要我继续重试?

📝 本章小结

本章学习了OpenClaw的自动化工作流功能：

核心功能

1. **定时任务设置**
   - 心跳机制原理
   - 简单提醒
   - 周报月报
   - Cron表达式

2. **网站监控实战**
   - 博客更新监控
   - 价格监控
   - 招聘信息监控
   - 监控频率优化

3. **日报自动推送**
   - 日报内容设计
   - AI行业日报
   - 技术资讯日报
   - 个人工作日报

4. **循环任务配置**
   - 固定间隔任务
   - 条件触发任务
   - 任务依赖管理
   - 错误重试机制

实战技巧

- ✅ 合理设置任务频率
- ✅ 选择合适的推送方式
- ✅ 设计有价值的日报内容
- ✅ 建立任务依赖关系
- ✅ 配置错误重试机制

下一步

- 学习第8章：Skills扩展
- 掌握ClawHub技能市场
- 学习自定义Skills开发
- 构建个人技能库

🎯 实战练习

练习1：设置早安提醒
1. 设置每天早上8点的早安提醒
2. 包含天气、日程、待办事项
3. 推送到飞书

练习2：监控技术博客
1. 选择3个你关注的技术博客
2. 设置更新监控
3. 有新文章时通知你

练习3：创建个人日报
1. 设计你的日报内容
2. 设置每天推送时间
3. 选择合适的推送方式

💡 常见问题

Q1：定时任务没有执行？
A：检查OpenClaw是否在后台运行，查看任务日志。

Q2：网站监控被封禁？
A：降低监控频率，添加随机延迟，使用代理。

Q3：日报内容不准确？
A：检查数据源，调整内容筛选规则。

Q4：循环任务占用资源？
A：优化任务逻辑，减少不必要的操作。

Q5：如何停止某个任务？
A：告诉OpenClaw"停止XX任务"即可。

```
---

**下一章预告**：第8章将学习Skills扩展，包括ClawHub技能市场、必装Skills推荐、自定义Skills开发等内容。
```

第8章 Skills扩展：让AI从"能说"到"能做"

> 💡 **本章目标**：深入理解Skills本质，学会使用ClawHub技能市场，掌握必装Skills推荐，学习自定义Skills开发和管理技巧。

> ⚠️ **重要提示**：没有Skills的OpenClaw只是1个聊天机器人，有了Skills才能真正成为你的AI助手！

> 📊 **Skills 生态概览**：OpenClaw 拥有庞大的 Skills 生态系统，包括内置49个、官方93个、社区1715+个，总计1800+个可用 Skills。详细分类和说明请参考 Skills 生态说明。

8.0 Skills本质：AI的"操作说明书"

8.0.1 什么是Skills?

简单理解：

Skills 本质上就是教 AI 按固定流程做事的操作说明书，一旦写好，就能像函数一样反复调用。

可以把 Skills 看成把"某类事情应该怎么专业做"这件事，封装成1个可复用、可自动触发的能力模块。

形象比喻：

把 AI 想象成1个**刚毕业的聪明但没经验的实习生**：

- **普通Prompt** = 你每次都要从头教他怎么做事（今天教一遍，明天还得重新教）
- **Rule / 记忆** = 你给他贴一张"公司行为守则"在工位上（一直生效，但只能管态度和格式）
- **MCP / Tools** = 你给他电脑装了一堆软件和API（他能调用外部工具，但不知道什么时候该用、怎么组合用）
- **Skills** = 你直接给他一整套**"岗位培训大礼包"**（PDF+流程图+SOP+话术模板+常用脚本），告诉他："当老板让你做这类事情时，就按这个文件夹里的方法来做"

8.0.2 Skills vs 传统Prompt

核心区别：按需加载 + 渐进式披露（只在需要时才把厚厚的 SOP 塞进上下文，极大节省 token）

对比项	普通 Prompt	Skills 机制
每次都要重新描述	是	否（只描述一次）
上下文长度占用	每次全量塞入	渐进式加载（只在触发时才读完整内容）
一致性	依赖每次 prompt 质量	高（固定 SOP + 模板）
复用性	手动复制粘贴	自动匹配 / slash 命令 / 项目共享
维护方式	改一次 prompt 就要重新发	修改 SKILL.md 文件，全局/项目生效

实际例子：

比如我们平时写文章，在没有 Skills 之前，每次都要按以下步骤重复说：

```
帮我总结文章 → 翻译 → 改成公众号风格 → 加标题 → 输出 Markdown
```

有了 Skills 之后：

```
你只需要一句：使用「技术文章转公众号」Skill

AI 会自动按你设定的步骤执行。
```

8.0.3 Skills vs MCP vs Tools

三者的区别：

特性	Skills	MCP	Tools
核心作用	知识复用	能力扩展	功能调用
实现方式	Markdown文件	服务器端配置	API接口
使用难度	⭐ 简单	⭐⭐⭐ 复杂	⭐⭐ 中等
Token消耗	低（渐进式加载）	高（启动时全量加载）	中等
适用场景	经验、最佳实践、工作流程	连接API、数据库、外部工具	单一功能调用
创建门槛	任何人都可以创建	需要编码能力	需要API开发
部署要求	无需服务器	需要服务器端设置	需要后端支持

使用建议：

- ✅ **Skills**：用于知识分享、经验复用、工作流程

- ✅ **MCP**：用于连接外部系统、数据库、API
- ✅ **Tools**：用于单一功能调用、快速集成

8.0.4 Skills的工作原理

渐进式加载机制：

Skills用渐进式加载来高效管理上下文：

1. **发现阶段**：启动时，AI 只加载每个技能的名称和描述，只保留最基本的识别信息
2. **激活阶段**：当任务匹配某个技能的描述时，AI 才把完整的 SKILL.md 指令读入上下文
3. **执行阶段**：AI 按照指令执行，按需加载参考文件或运行代码

这种设计让 AI 保持快速，同时能按需获取更多信息。

Token消耗对比：

```
传统方式（全量加载）：
- 10个Skills × 500字/个 = 5000字
- 约1250 tokens

Skills方式（渐进式加载）：
- 发现阶段：10个Skills × 50字/个 = 500字（约125 tokens）
- 激活阶段：只加载需要的1个Skills = 500字（约125 tokens）
- 总计：约250 tokens

节省：80% tokens
```

8.0.5 支持Skills的工具

目前能用 Skills 的主流客户端：

排序	工具名	是否免费使用Skills	推荐人群	技能存放默认路径	备注
1	Claude Code	是（官方）	所有人	~/.claude/skills	标准制定者，生态最全
2	Cursor	是	写代码最常用	~/.cursor/skills	几乎无缝兼容Claude Skills
3	Trae / OpenCode	是	追求性价比	看工具设置	国内用户较多
4	VS Code + 插件	部分支持	已经深度用vscode	插件设置里配置	正在快速跟进
5	扣子/其他国内平台	部分支持	喜欢网页版	平台自带技能市场	有的要会员

OpenClaw的优势：

- ✅ 完全兼容Claude Skills格式
- ✅ 支持本地和云端部署
- ✅ 可以通过飞书、企微、钉钉使用
- ✅ 支持自定义Skills开发
- ✅ 社区生态丰富（1715+个Skills）

8.0.6 Skills核心结构

最小Skills结构：

1个 Skill 本质上就是1个 Markdown 文件（文件名固定为 SKILL.md）

```
my-skill/
└── SKILL.md     （唯一必需）
```

SKILL.md 基本模板：

```
---
name: pdf-processing
description: 从 PDF 中提取文本和表格，填写表单，并合并文档
---

# PDF 处理

## 使用场景

当需要对 PDF 文件进行操作时使用，例如：
- 提取 PDF 文本或表格数据
- 填写 PDF 表单
- 合并多个 PDF 文件

## 提取文本

- 使用 `pdfplumber` 提取文本型 PDF 内容
- 扫描版 PDF 需配合 OCR 工具

## 填写表单

- 读取 PDF 表单字段
- 按输入数据填充并生成新文件
```text
**Skills核心结构示意图**：

![Skills核心结构图 - SKILL.md定义技能能力]
(https://upload.maynor1024.live/file/1770813820310_image1.png)

**Frontmatter字段说明**：

| 字段 | 必需 | 说明 |
|------|------|------|
| name | ✅ | Skill 名称，最长 64 字符，只能使用小写字母、数字和 `-`，且不能以 `-` 开头或结尾 |
| description | ✅ | 功能与使用场景说明，最长 1024 字符，不能为空 |
| license | ❌ | 许可证名称或指向随 Skill 附带的许可证文件 |
| compatibility | ❌ | 环境与依赖说明（产品、系统包、网络权限等），最长 500 字符 |
| metadata | ❌ | 自定义键值对，用于扩展元数据（如作者、版本号） |
| allowed-tools | ❌ | 允许使用的工具列表（空格分隔，实验性功能） |

**完整Skills结构**（包含资源文件）：
```
```

```
如果你需要一些参考资料、参考实例、执行脚本，可以使用更复杂的 Skill 目录结构：
```

```
my-skill/
├── SKILL.md # 必需：指令 + 元数据
├── scripts/ # 可选：可执行代码
├── references/ # 可选：文档资料
└── assets/ # 可选：模板、资源
```

```
![复杂Skills目录结构 - 包含脚本和依赖]
(https://upload.maynor1024.live/file/1770813829999_image2.png)

---

## 🎯 Skills生态概览

**最新数据（2026年2月）**：
- 📊 **总技能数**：1715+个技能
- 📁 **主要分类**：31个分类
- 🌟 **活跃贡献者**：数百位开发者
- 📈 **增长速度**：每月新增50+个技能

**主要分类及技能数量**：
- 🌐 Web前端开发：46个
- 💻 编程代理和IDE：55个
- 🔧 Git和GitHub：34个
- ☁️ DevOps和云服务：144个
- 🌐 浏览器和自动化：69个
- 🎨 图像和视频生成：41个
- 🤖 AI和LLMs：159个
- 📊 数据和分析：18个
- ✅ 生产力和任务管理：93个

> 💡 **Skills是什么？** Skills是OpenClaw的插件系统，是让AI助手从"能说"到"能做"的关键转变。安装了Skills之后，你的AI助手可以自动浏览网页、收发邮件、操作GitHub、控制智能家居、生成图片、管理任务，甚至控制你的Tesla汽车！

## 🔧 本章内容

- 8.1 ClawHub技能市场
- 8.2 核心Skills推荐
- 8.3 Skills 安装方法
- 8.4 实战应用案例
- 8.5 安全使用指南
- 8.6 Skills开发指南
- 8.7 Skills管理技巧

---

## 8.1 ClawHub技能市场
```

```
### 8.1.1 什么是ClawHub

**定义**：
ClawHub是OpenClaw的官方技能市场（https://clawhub.com），类似于App Store，提供各种扩展功能。

**核心价值**：
- 🎯 **扩展能力**：让OpenClaw能做更多事情
- 🚀 **快速部署**：一键安装，即刻使用
- 🌍 **社区驱动**：开发者共享优质Skills
- 🔄 **持续更新**：Skills不断优化升级
- 📦 **统一管理**：安装、更新、备份一站式

**与其他AI的区别**：

| 特性 | OpenClaw + Skills | ChatGPT Plugins | Claude |
|------|------------------|----------------|--------|
| 本地执行 | ✅ | ❌ | ❌ |
| 系统操作 | ✅ | ❌ | ❌ |
| 自定义开发 | ✅ | ❌ | ❌ |
| 社区生态 | ✅ | ✅ | ❌ |
| 免费使用 | ✅ | 部分付费 | ❌ |

### 8.1.2 Skills加载机制

**Skills加载位置**：

OpenClaw从3个位置加载Skills，优先级从高到低：
```

1. 工作区Skills：/skills（最高优先级）
   - 单智能体专用
   - 项目特定的Skills
2. 托管/本地Skills：~/.openclaw/skills
   - 所有智能体共享
   - 用户自定义Skills
3. 内置Skills：随安装包发布（最低优先级）
   - OpenClaw官方Skills
   - 基础功能Skills

```
**优先级规则**：
```

如果同名Skills存在于多个位置：
工作区Skills > 托管/本地Skills > 内置Skills

示例：

- 内置Skills：file-search v1.0
- 本地Skills：file-search v1.2（自定义版本）
- 工作区Skills：file-search v2.0（项目专用）

最终加载：工作区的 v2.0 版本

额外Skills目录：

可以通过配置添加额外的Skills文件夹：

```json
{
  "skills": {
    "load": {
      "extraDirs": [
        "/path/to/shared-skills",
        "/path/to/team-skills"
      ]
    }
  }
}
```text
### 8.1.3 ClawHub使用指南

**访问方式**：

```bash
# 方式1：网页访问（推荐）
https://clawhub.com

# 方式2：命令行工具
clawhub --help

# 方式3：OpenClaw内访问
你：打开ClawHub
```text
**常用命令**：

```bash
# 搜索Skills
clawhub search <关键词>

# 查看Skills详情
clawhub info <skill-slug>

# 安装Skills到工作区
clawhub install <skill-slug>
```

```
# 安装到指定目录
clawhub install <skill-slug> --dir /path/to/skills

# 更新单个Skills
clawhub update <skill-slug>

# 更新所有Skills
clawhub update --all

# 同步（扫描+发布更新）
clawhub sync --all

# 列出已安装的Skills
clawhub list

# 卸载Skills
clawhub uninstall <skill-slug>
```text
**安装位置说明**：

```bash
# 默认安装到当前工作目录
clawhub install nano-banana-pro
# 安装到：./skills/nano-banana-pro/

# 如果不在工作目录，回退到配置的工作区
# 安装到：<workspace>/skills/nano-banana-pro/
```text
**ClawHub界面布局**：


![ClawHub技能市场 - 浏览和安装技能]
(https://upload.maynor1024.live/file/1770738350555_image-20260210234513022.png)

### 8.1.3 技能分类和搜索

**主要分类**：

**1. 文件管理类**
- 文件搜索
- 文件整理
- 批量处理
- 格式转换
```
```

```
**2. 知识管理类**
- 网页剪藏
- 笔记同步
- 文献管理
- 知识图谱

**3. 日程管理类**
- 日历同步
- 提醒设置
- 任务管理
- 时间追踪

**4. 自动化类**
- 定时任务
- 网站监控
- 数据采集
- 流程自动化

**5. 工具类**
- 截图工具
- 翻译助手
- 计算器
- 单位转换

**6. 娱乐休闲类**
- 音乐播放
- 天气查询
- 新闻阅读
- 游戏娱乐

**搜索技巧**：
```

按名称搜索

搜索：file-search

按功能搜索

搜索：文件搜索

按标签搜索

搜索：#文件管理 #效率工具

按作者搜索

搜索：@作者名

组合搜索

搜索：文件 #管理 @官方

```
### 8.1.4 技能评价体系

**评价维度**：

| 维度 | 说明 | 权重 |
|------|------|------|
| ⭐ 评分 | 用户评分（1-5星） | 30% |
| 📥 下载量 | 安装次数 | 20% |
| 🔄 更新频率 | 维护活跃度 | 20% |
| 📝 文档质量 | 说明完善度 | 15% |
| 🐛 Bug数量 | 稳定性 | 15% |

**评分标准**：
```

⭐⭐⭐⭐⭐ (5.0) - 完美

- 功能完善
- 文档详细
- 无明显Bug
- 持续更新

⭐⭐⭐⭐ (4.0-4.9) - 优秀

- 功能完整
- 文档清晰
- 偶尔小Bug
- 定期更新

⭐⭐⭐ (3.0-3.9) - 良好

- 基本功能可用

- 文档一般
- 有些Bug
- 更新较慢

⭐⭐ (2.0-2.9) - 一般

- 功能不完整
- 文档缺失
- Bug较多
- 很少更新

⭐ (1.0-1.9) - 较差

- 功能有问题
- 无文档
- Bug很多
- 不再维护

```
**如何选择Skills**：
```

✅ 推荐安装：

- 评分 ≥ 4.0
- 下载量 > 1000
- 最近3个月有更新
- 文档完善

⚠️ 谨慎安装：

- 评分 3.0-3.9
- 下载量 < 1000
- 更新不频繁
- 文档不全

❌ 不推荐：

- 评分 < 3.0
- 长期不更新
- 无文档
- Bug多

```
### 8.1.5 Skills 安装和管理

**安装方式**：

**方式1：通过ClawHub安装**
```

1. 打开ClawHub
2. 搜索Skills
3. 点击"安装"按钮
4. 等待安装完成

```
**方式2：通过命令行安装**
```bash
# 安装单个Skill
openclaw skill install file-search

# 安装多个Skills
openclaw skill install file-search note-sync calendar-sync

# 从URL安装
openclaw skill install https://github.com/user/skill-name
```text
**方式3：通过OpenClaw对话安装**
```

你：帮我安装file-search这个Skill

OpenClaw：好的，我来帮你安装！

正在安装 file-search...

- 下载中... ✅
- 解压中... ✅
- 配置中... ✅
- 测试中... ✅

安装成功！ ✅

Skill信息：

- 名称：File Search
- 版本：v1.2.0
- 作者：OpenClaw官方
- 功能：智能文件搜索

现在你可以使用文件搜索功能了！

查看已安装Skills：

```bash
# 列出所有已安装的Skills
openclaw skill list

# 输出示例：
已安装Skills（12个）：

文件管理类（3个）：
✅ file-search v1.2.0
✅ file-organizer v2.0.1
✅ batch-processor v1.5.0

知识管理类（4个）：
✅ web-clipper v1.8.0
✅ note-sync v2.1.0
✅ github-manager v1.3.0
✅ paper-reader v1.0.5

日程管理类（2个）：
✅ calendar-sync v3.0.0
✅ reminder v1.4.0

工具类（3个）：
✅ screenshot v2.2.0
✅ translator v1.6.0
✅ calculator v1.1.0
```text
**更新Skills**：

```bash
# 检查更新
openclaw skill update --check

# 更新所有Skills
openclaw skill update --all

# 更新指定Skill
openclaw skill update file-search

# 输出示例：
```

```
检查更新中...

发现3个可更新的Skills：
1. file-search: v1.2.0 → v1.3.0
2. note-sync: v2.1.0 → v2.2.0
3. calendar-sync: v3.0.0 → v3.1.0

是否更新？[Y/n]
```text
**卸载Skills**：

```bash
# 卸载单个Skill
openclaw skill uninstall file-search

# 卸载多个Skills
openclaw skill uninstall file-search note-sync

# 输出示例：
正在卸载 file-search...
- 停止服务... ✅
- 清理配置... ✅
- 删除文件... ✅

卸载成功！✅
```text
---

## 8.2 必装Skills推荐

### 8.2.1 文件管理类Skills

**1. file-search（智能文件搜索）**
```

⭐ 评分：5.0/5.0
📥 下载量：50,000+
👤 作者：OpenClaw官方

功能：

- 基于内容的智能搜索
- 支持多种文件格式
```

- 快速索引和检索
- 模糊匹配

使用示例：
你：搜索包含"发票"的PDF文件

OpenClaw：找到3个匹配文件：

1. 2026年1月发票.pdf
2. 跑步机购买发票.pdf
3. 办公用品发票.pdf

```
**2. file-organizer（文件自动整理）**
```

⭐ 评分：4.8/5.0
📥 下载量：35,000+
👤 作者：OpenClaw官方

功能：

- 智能分类文件
- 批量重命名
- 重复文件检测
- 自动归档

使用示例：
你：整理下载文件夹

OpenClaw：正在整理...

- 图片 → Pictures/Downloads/
- 文档 → Documents/Downloads/
- 视频 → Videos/Downloads/
- 其他 → Others/Downloads/

整理完成！共处理156个文件 ✅

```
**3. batch-processor（批量文件处理）**
```

⭐ 评分：4.7/5.0
📥 下载量：28,000+
👤 作者：社区开发者

功能：

- 批量格式转换
- 批量压缩
- 批量提取信息
- 批量重命名

使用示例：
你：把这个文件夹里的所有PNG转成JPG

OpenClaw：正在转换...

- image1.png → image1.jpg ✅
- image2.png → image2.jpg ✅
- image3.png → image3.jpg ✅
  ...
  共转换25个文件 ✅

```
### 8.2.2 知识管理类Skills

**4. web-clipper（网页剪藏）**
```

⭐ 评分：4.9/5.0
📥 下载量：45,000+
👤 作者：OpenClaw官方

功能：

- 网页内容抓取
- 智能摘要生成
- 多平台保存（备忘录/Notion/Obsidian）
- 标签管理

使用示例：
你：保存这篇文章到备忘录
https://example.com/article

OpenClaw：已保存！
标题：AI技术发展趋势
摘要：[自动生成的摘要]
标签：#AI #技术 #趋势
位置：备忘录 > AI学习 ✅

```
**5. note-sync（笔记同步）**
```

⭐ 评分：4.8/5.0
📥 下载量：32,000+
👤 作者：OpenClaw官方

功能：

- 多平台笔记同步
- 实时备份
- 版本控制
- 冲突解决

支持平台：

- Mac备忘录
- Notion
- Obsidian
- Evernote

```
**6. github-manager（GitHub项目管理）**
```

⭐ 评分：4.6/5.0
📥 下载量：25,000+
👤 作者：社区开发者

功能：

- 项目信息分析
- README总结
- 代码结构理解
- Star/Fork追踪

使用示例：
你：分析这个GitHub项目
https://github.com/user/project

OpenClaw：分析完成！
项目：Awesome Project
Stars：15.3k
语言：TypeScript
核心功能：[详细分析]
推荐指数：⭐⭐⭐⭐⭐

```
### 8.2.3 日程管理类Skills

**7. calendar-sync（日历同步）**
```

⭐ 评分：5.0/5.0
📥 下载量：60,000+
👤 作者：OpenClaw官方

功能：

- Mac日历集成
- 自动创建事件
- 智能提醒
- 跨设备同步

使用示例：
你：明天下午3点，会议室A开会

OpenClaw：已创建日历事件！
标题：会议
时间：2026-02-12 15:00-16:00
地点：会议室A
提醒：提前15分钟 ✅

```
**8. reminder（智能提醒）**
```

⭐ 评分：4.7/5.0
📥 下载量：38,000+
👤 作者：OpenClaw官方

功能：

- 多级提醒设置
- 智能提醒策略
- 位置提醒
- 重复提醒

使用示例：
你：每天早上8点提醒我喝水

OpenClaw：已设置提醒！
内容：该喝水了💧

时间：每天 08:00
方式：飞书消息 ✅

```
### 8.2.4 自动化类Skills

**9. scheduler（任务调度）**
```

⭐ 评分：4.9/5.0
📥 下载量：42,000+
👤 作者：OpenClaw官方

功能：

- 定时任务设置
- Cron表达式支持
- 任务依赖管理
- 错误重试

使用示例：
你：每天早上8点推送AI日报

OpenClaw：已设置定时任务！
任务：AI日报推送
时间：每天 08:00
Cron：0 8 * * *
状态：已启用 ✅

```
**10. monitor（网站监控）**
```

⭐ 评分：4.6/5.0
📥 下载量：28,000+
👤 作者：社区开发者

功能：

- 网站变化检测
- 价格监控
- 内容更新提醒
- RSS订阅

使用示例：
你：监控这个博客的更新
https://example-blog.com

OpenClaw：已设置监控！
网站：Example Blog
频率：每天2次
通知：飞书消息 ✅

```
### 8.2.5 工具类Skills

**11. screenshot（截图工具）**
```

⭐ 评分：4.8/5.0
📥 下载量：35,000+
👤 作者：OpenClaw官方

功能：

- 全屏截图
- 窗口截图
- 区域截图
- OCR识别

使用示例：
你：截图当前屏幕

OpenClaw：已截图！
[发送截图]
是否需要OCR识别文字？

```
![截图技能演示 - 自动捕获屏幕内容]
(https://upload.maynor1024.live/file/1770176353570_image_35.jpg)

**12. translator（翻译助手）**
```

⭐ 评分：4.7/5.0
📥 下载量：30,000+
👤 作者：社区开发者

功能：

- 多语言翻译
- 实时翻译
- 文档翻译
- 术语库管理

使用示例：
你：把这段话翻译成英文
"人工智能正在改变世界"

OpenClaw：翻译结果：
"Artificial Intelligence is changing the world"

8.2.6 Skills推荐总结

必装Top 10：

排名	Skill	分类	评分	推荐理由
1	calendar-sync	日程管理	5.0	日历集成必备
2	file-search	文件管理	5.0	文件搜索神器
3	web-clipper	知识管理	4.9	知识收集利器
4	scheduler	自动化	4.9	定时任务必备
5	note-sync	知识管理	4.8	笔记同步工具
6	file-organizer	文件管理	4.8	文件整理助手
7	screenshot	工具	4.8	截图OCR工具
8	reminder	日程管理	4.7	智能提醒系统
9	batch-processor	文件管理	4.7	批量处理工具
10	translator	工具	4.7	翻译助手

安装建议：

```bash
# 基础套装（必装）
openclaw skill install calendar-sync file-search web-clipper

# 进阶套装（推荐）
openclaw skill install scheduler note-sync file-organizer

# 完整套装（全能）
openclaw skill install calendar-sync file-search web-clipper \
  scheduler note-sync file-organizer screenshot reminder \
  batch-processor translator
```text
### 8.2.7 核心Skills详解（2026年必备）

> 💡 **新增内容**：基于1715+个技能生态，精选7大核心技能，每个都经过实战验证。

#### 1. McPorter——跨平台连接基石 🏗️

**核心作用**：
让OpenClaw支持MCP（Model Context Protocol）协议，无需编写胶水代码，直接连接成千上万个现成的MCP Server。

**支持平台**：

- PostgreSQL数据库
- GitHub
- Slack
- Notion
- 其他主流平台

**安装命令**：
```bash
npx clawhub@latest install mcporter
```text
**配置示例**：
```bash
# 配置MCP服务器（以连接本地文件为例）
openclaw mcp add --transport stdio local-files npx -y
@modelcontextprotocol/server-filesystem /root/Documents
```text
**使用场景**：
- "读取Notion中的项目文档，整理成Markdown"
- "把GitHub上的最新代码提交记录同步到本地"

**真实效果**：某开发团队使用McPorter技能，将日常协作效率提升了3倍。

#### 2. Brave Search——实时信息检索 🔍

**核心作用**：
解决传统AI Agent"数据过时"的问题，让OpenClaw能进行实时全网搜索，获取最新的GitHub Issue、StackOverflow解答、行业资讯。

**安装命令**：
```bash
npx clawhub@latest install brave-search
```text
**使用场景**：
- **代码报错排查**："帮我排查这个Python报错的原因，找最新的解决方案"
- **竞品调研**："查一下某产品最新功能的实现方式，附代码片段"

**效果**：2分钟即可得到带参考链接的详细报告，告别"凭训练数据瞎猜"的时代。

#### 3. TranscriptAPI——视频知识提取 🎥

**核心作用**：
稳定抓取YouTube视频字幕，带精确时间戳，将视频中的知识转化为可编辑的文本。

**安装命令**：
```bash
npx clawhub@latest install transcript-api
```text
**使用场景**：
"提取这个2小时Next.js教程视频的核心代码逻辑，按章节整理成学习笔记"

**价值**：无需手动拉进度条，AI直接将知识"喂到嘴边"，学习效率提升3倍以上。

#### 4. File System Manager——本地文件处理 💾

**核心作用**：
赋予OpenClaw本地文件的读写、修改、重构权限，支持批量修改代码、修复语法错误、自动提交Git。

**安装命令**：
```bash
npx clawhub@latest install file-system-manager
```text
![1Password CLI集成 - 安全管理密码凭证]
(https://upload.maynor1024.live/file/1770778507814_image-20260211105458447.png)



**重要安全配置**：

```bash
# 配置授权目录（仅开放工作目录，避免全硬盘访问）
openclaw config set fs.allow-path /root/Projects
```text
**使用场景**：
- "帮我重构这个React组件，优化代码结构并修复ESLint报错"
- "将本地Markdown文件转为PDF，保存到指定目录"

**注意**：该技能是双刃剑，需严格控制访问目录，避免误操作。

#### 5. Headless Browser (Playwright)——浏览器自动化 🤖

**核心作用**：
模拟真实人类的浏览器操作，支持点击、输入、截图、表单提交，针对无API的老旧网站实现自动化操作。

**安装命令**：
```bash
npx clawhub@latest install playwright-headless
```

```text
**使用场景**：
- "每天早上8点自动登录公司抢票系统，帮我预约车票"
- "定时截图某政府网站的公告，有更新就保存并提醒"

**注意**：该功能过于强大，需合规使用，避免违反平台规则。

#### 6. Design-Doc-Mermaid——图表自动生成 📊

**核心作用**：
通过自然语言指令生成Mermaid代码，自动渲染架构图、时序图、流程图。

**安装命令**：
```bash
npx clawhub@latest install design-doc-mermaid
```text
**使用场景**：
"帮我画1个用户注册的时序图，包含前端、后端、数据库交互"

**效果**：AI直接生成Mermaid代码并渲染成图，告别用画图工具手动拖拽的时代。

#### 7. Google Workspace集成——办公自动化 📧

**核心作用**：
无缝连接Gmail、Google Calendar、Google Docs，实现邮件整理、日程同步、文档自动生成。

**安装命令**：
```bash
npx clawhub@latest install google-workspace
```text
**授权配置**：
```bash
# 授权Google账号（按终端提示完成浏览器认证）
openclaw auth google
```text
**使用场景**：
- "查一下我这周的Gmail邮件和Calendar日程，生成一份简洁的周报，发给老板"
- "根据会议纪要，自动创建Google Calendar日程，邀请参会人员"

#### 8. find-skills + ProactiveAgent——Skills双幻神 🌟🌟

**核心作用**：
- **find-skills**：智能发现Skills，OpenClaw遇到问题时自动寻找合适的Skills
```

- **ProactiveAgent**：主动预测需求，观察使用习惯后主动提出自动化建议

安装命令：
```bash
# 安装Skills双幻神
npx clawhub@latest install find-skills
npx clawhub@latest install proactive-agent
```text
**使用场景**：
- find-skills：当OpenClaw无法完成某个任务时，自动搜索并推荐合适的Skills
- ProactiveAgent：做了几次日报转HTML后，主动问"要不要我帮你自动化这个流程？"

**GitHub链接**：
- find-skills: https://github.com/vercel-labs/skills/tree/main/skills/find-skills
- ProactiveAgent: https://github.com/leomariga/ProactiveAgent

⚠️ **安全提示**：ProactiveAgent安装时可能显示VirusTotal警告（因包含外部API调用），这是正常的，可以安全使用。

**核心Skills 安装命令汇总**：
```bash
# 一键安装9大核心Skills（包含Skills双幻神）
npx clawhub@latest install mcporter brave-search transcript-api \
  file-system-manager playwright-headless design-doc-mermaid google-workspace \
  find-skills proactive-agent
```text
---

## 8.3 Skills 安装方法详解

> 💡 **三种安装方式**：ClawHub CLI（推荐）、对话式安装（最简单）、手动安装（高级用户）

### 8.3.1 方法一：ClawHub CLI一键安装（推荐）⭐

**特点**：
- ✅ 最简单、最快捷
- ✅ 自动处理依赖
- ✅ 支持版本管理

**安装命令**：
```bash
npx clawhub@latest install <技能名称>
```text

```
实际示例：
```bash
# 安装网页搜索技能
npx clawhub@latest install brave-search

# 安装多个技能
npx clawhub@latest install brave-search transcript-api file-system-manager
```text
**常用管理命令**：
```bash
# 查看已安装技能
npx clawhub@latest list

# 更新技能
npx clawhub@latest update <skill-slug>

# 卸载技能
npx clawhub@latest uninstall <skill-slug>

# 更新所有技能
npx clawhub@latest update --all
```text
### 8.3.2 方法二：对话式安装（最懒人方式）

**特点**：
- ✅ 无需命令行
- ✅ 适合新手
- ✅ AI自动配置

**使用方法**：

直接在聊天界面中输入：
```

请安装这个技能：https://github.com/openclaw/skills/tree/main/skills/steipete/brave-search

AI会自动帮你下载和配置，完全不需要手动操作！

适用场景：
- 不熟悉命令行的用户
- 快速测试某个技能
- 临时安装单个技能

8.3.3 方法三：手动安装（高级用户）

特点：
- ✅ 完全掌控
- ✅ 可自定义配置
- ✅ 适合开发调试

安装步骤：
```bash
# 1. 克隆技能仓库
git clone https://github.com/openclaw/skills

# 2. 复制到全局技能目录
cp -r skills/skills/<作者>/<技能名> ~/.openclaw/skills/

# 3. 重启OpenClaw
openclaw gateway restart
```text
---

## 8.4 实战应用案例

> 💡 **三大场景完整案例**：内容创作、团队协作、智能家居，都有真实数据支撑。

### 8.4.1 场景一：自动化内容创作流程 ✍️

**需求背景**：
自媒体博主每天需要发布3篇公众号文章，包括搜索资料、撰写内容、配图、排版，传统方式需要6小时/天。

**使用的Skills组合**：
1. `brave-search` - 搜索最新资讯
2. `deep-research` - 深度研究主题
3. `fal-ai` - 生成配图
4. `notion` - 保存草稿
5. `markdown-formatter` - 格式化文章

**效果对比**：

| 指标 | 传统方式 | 使用 OpenClaw Skills |
|------|---------|-------------------|
| ⏰ 时间消耗 | 6小时/天 | 2小时/天 |
| 📈 文章产量 | 3篇/天 | 5篇/天 |
| 💰 配图成本 | 300元/天 | 0元/天 |

**具体操作流程**：
1. **信息收集**：使用brave-search搜索当日热点话题
2. **深度研究**：通过deep-research对选定话题进行深入分析
3. **内容生成**：AI根据研究成果自动撰写文章草稿
4. **配图生成**：使用fal-ai生成原创配图，避免版权问题
5. **格式排版**：自动格式化为公众号要求的样式

### 8.4.2 场景二：开发团队协作自动化 💻

**需求背景**：
10人开发团队需要管理日常协作，包括代码审查、任务分配、进度跟踪等。

**使用的Skills组合**：
1. `github` - 代码仓库管理
2. `linear` - 任务分配
3. `slack` - 团队通知
4. `google-calendar` - 会议安排

**效果提升**：
- 📊 PR审查时间：从2天降到4小时
- 🎯 任务分配效率：提升80%
- 💬 沟通成本：减少50%
- 📅 会议安排：自动化100%

**安装命令**：
```bash
npx clawhub@latest install github linear-integration slack-bot google-workspace
```text
**自动化工作流**：
1. **自动代码审查**：每次PR提交自动进行基础代码检查
2. **任务自动分配**：根据团队成员工作量和专长智能分配任务
3. **进度自动同步**：每日自动生成项目进度报告并发送到Slack
4. **会议自动安排**：根据团队成员日历自动安排最佳会议时间

### 8.4.3 场景三：智能家居全自动化 🏠

**需求背景**：
根据天气、时间、位置自动控制家中所有设备，提升生活品质。

**使用的Skills组合**：
1. `home-assistant` - 智能家居控制
2. `weather-api` - 天气查询
3. `location-tracker` - 位置追踪
4. `automation-scheduler` - 自动化调度

**实现的智能场景**：
- 🌅 **早晨唤醒**：早上7点自动打开窗帘、启动咖啡机
- 🌧️ **天气适应**：下雨时自动关闭窗户
- 🚗 **离家模式**：离家10分钟自动关闭所有灯光和空调
- 🏠 **回家预热**：到家前5分钟自动打开空调和灯光

**安装命令**：
```bash
npx clawhub@latest install home-assistant weather-api location-tracker automation-scheduler
```text
---

## 8.5 安全使用指南

> ⚠️ **重要**：Skills虽然强大，但也存在安全风险。必须严格遵循安全原则。

### 8.5.1 核心安全原则

1. **隔离运行环境**：重要的OpenClaw实例建议运行在Docker或专用虚拟机中
2. **严控权限范围**：对高危Skill仅开放必要的工作目录/操作权限
3. **优先官方认证**：安装带"官方认证""高星标"的技能
4. **安装前检查代码**：查看技能代码确认无恶意逻辑

### 8.5.2 安全配置命令

**权限控制**：
```bash
# 查看Skill的核心代码，确认无恶意逻辑
clawhub view file-system-manager

# 限制OpenClaw的系统资源使用
```

```
docker update --cpus=1 --memory=2g openclaw-2026

# 备份OpenClaw 配置，防止恶意修改
docker cp openclaw-2026:/root/.openclaw /root/openclaw-backup
```text
**安全检查清单**：
- ✅ 查看技能的GitHub仓库
- ✅ 检查技能的依赖项
- ✅ 阅读技能的权限要求
- ✅ 使用skill-scanner扫描恶意代码
- ✅ 在测试环境中先试用

### 8.5.3 故障排查

**问题一：技能安装失败**
```bash
# 网络超时问题：检查服务器网络连接
ping github.com

# 配置国内镜像源（如遇网络问题）
npm config set registry https://registry.npmmirror.com
```text
**问题二：技能加载失败**
```bash
# 查看技能加载状态
openclaw plugins list

# 重新加载技能
openclaw plugins load <skill-name>

# 更新所有技能
clawhub update --all
openclaw gateway restart
```text
**问题三：技能执行无响应**
```bash
# 查看技能执行日志
openclaw logs --skill <skill-name>

# 检查权限配置
openclaw config get fs.allow-path
```text
---
```
```

```
## 8.6 Skills开发指南

> 💡 **创建自定义Skills**：如果现有的1715+个技能无法满足你的特殊需求，可以创建自己的技能。

### 8.6.1 创建自定义Skills

**创建步骤**：
1. **创建技能文件夹**：在 `~/.openclaw/skills/` 中创建新文件夹
2. **编写描述文件**：创建 `SKILL.md` 描述文件
3. **添加脚本配置**：编写必要的脚本和配置文件
4. **测试技能功能**：在本地测试技能是否正常工作
5. **分享到社区**：将技能贡献给社区

**技能目录结构**：
```

```
my-custom-skill/
├── SKILL.md # 技能描述文档
├── config.json # 配置文件
├── scripts/ # 执行脚本
│ └── main.js
└── README.md # 使用说明
```

```
**项目文件结构示例**：

![自定义Skills项目结构 - 完整的开发示例]
(https://upload.maynor1024.live/file/1770813835155_image3.png)

你的项目现在看起来应该是这样的：
```

```
my-project/
├── src/
│ └── test.py # 项目源码
├── .claude/
│ ├── skills/
│ │ └── hello-world/
│ │ ├── SKILL.md # Skill 定义（YAML + Instructions，机器可执行）
│ │ └── README.md # Skill 说明（人类阅读，可选）
│ └── config.yml # Claude 项目级配置（可选）
├── .gitignore
└── README.md # 项目整体说明
```

```
**Claude Code执行效果**：

接下来我们在终端执行以下命令启动 Claude Code：

```bash
claude
```text
输入任务：
```

帮我写1个计算用户折扣的函数

Claude 会扫描已安装的 Skills，发现你的请求涉及 "Python 代码编写"，匹配了 python-naming-standard。

![Claude Code执行效果 - Skills自动匹配和调用](https://upload.maynor1024.live/file/1770813836904_image4.png)

它会根据 SKILL.md 中的要求，生成如下代码：

```python
def _internal_get_discount(user_score):
    # 计算逻辑...
    return discount
```text
### 8.6.2 技能开发最佳实践

**设计原则**：
1. **单一职责**：每个技能只解决1个特定问题
2. **接口简单**：提供清晰简单的使用接口
3. **错误处理**：完善的错误处理和用户提示
4. **文档完整**：提供详细的使用文档和示例

**示例技能模板**：
```javascript
// 简单技能示例
module.exports = {
  name: "my-custom-skill",
  description: "这是我的自定义技能",
  version: "1.0.0",

  async execute(params, context) {
    try {
      // 技能核心逻辑
      const result = await doSomething(params);
      return { success: true, data: result };
    } catch (error) {
      return { success: false, error: error.message };
    }
  }
};
```text
---

## 8.7 自定义Skills开发（原8.3节）

### 8.3.1 Skills开发基础

**为什么要开发Skills**：

1. **满足个性化需求**
   - 官方Skills无法满足
   - 特定业务场景
   - 企业内部工具

2. **学习和成长**
   - 深入理解OpenClaw
   - 提升编程能力
   - 贡献开源社区

3. **商业价值**
   - 开发付费Skills
   - 提供定制服务
   - 建立个人品牌

**开发前准备**：

```bash
# 1. 安装开发工具
npm install -g openclaw-cli

# 2. 创建开发环境
openclaw dev init

# 3. 学习文档
openclaw docs

# 4. 查看示例
openclaw examples
```text
### 8.3.2 Skills文件格式（AgentSkills兼容）

**基本格式**：

OpenClaw使用兼容AgentSkills的Skills文件夹。每个Skills是1个包含`SKILL.md`的目录。

**最小示例**：

```markdown
---
name: nano-banana-pro
description: Generate or edit images via Gemini 3 Pro Image
---

# Nano Banana Pro

这个Skills可以生成和编辑图片。

## 使用方法

发送：画一只可爱的猫
```text
**完整示例**：

```markdown
---
name: gemini
description: Use Gemini CLI for coding assistance and Google search lookups.
homepage: https://github.com/example/gemini-skill
user-invocable: true
disable-model-invocation: false
metadata: {
  "openclaw": {
    "emoji": "♊",
    "requires": {
      "bins": ["gemini"],
      "env": ["GEMINI_API_KEY"]
    },
    "primaryEnv": "GEMINI_API_KEY",
    "install": [
      {
        "id": "brew",
        "kind": "brew",
        "formula": "gemini-cli",
        "bins": ["gemini"],
        "label": "Install Gemini CLI (brew)"
      }
    ]
  }
}
```

```
---

# Gemini Skills

使用Gemini CLI进行编程辅助和Google搜索。

## 功能

- 代码生成
- 问题解答
- 网络搜索

## 使用方法

发送：用Gemini搜索最新的AI新闻
```text
**Frontmatter字段说明**：

| 字段 | 必填 | 说明 |
|------|------|------|
| `name` | ✅ | Skills名称（唯一标识） |
| `description` | ✅ | Skills描述 |
| `homepage` | ❌ | 项目主页URL |
| `user-invocable` | ❌ | 是否作为斜杠命令暴露（默认true） |
| `disable-model-invocation` | ❌ | 是否从模型提示词中排除（默认false） |
| `command-dispatch` | ❌ | 命令调度模式（tool=直接调度到工具） |
| `command-tool` | ❌ | 要调用的工具名称 |
| `command-arg-mode` | ❌ | 参数模式（raw=原始字符串） |
| `metadata` | ❌ | 元数据（单行JSON对象） |

**注意事项**：
```

⚠️ 重要：

1. 内嵌智能体的解析器仅支持单行frontmatter键
2. metadata必须是单行JSON对象
3. 在说明中使用{baseDir}引用Skills文件夹路径
```

8.3.3 Skills门控（加载时过滤）

什么是门控：

门控是指在加载时根据条件过滤Skills，只加载满足条件的Skills。

门控配置：

通过`metadata.openclaw`配置门控条件：

```markdown
---
name: nano-banana-pro
description: Generate or edit images
metadata: {
  "openclaw": {
    "requires": {
      "bins": ["uv"],
      "env": ["GEMINI_API_KEY"],
      "config": ["browser.enabled"]
    },
    "primaryEnv": "GEMINI_API_KEY",
    "os": ["darwin", "linux"]
  }
}
---
```text
**门控字段说明**：

| 字段 | 说明 | 示例 |
|------|------|------|
| `always` | 始终包含（跳过其他门控） | `"always": true` |
| `os` | 限制操作系统 | `"os": ["darwin", "linux"]` |
| `requires.bins` | 必需的二进制文件（全部） | `"bins": ["uv", "python"]` |
| `requires.anyBins` | 必需的二进制文件（任一） | `"anyBins": ["npm", "yarn"]` |
| `requires.env` | 必需的环境变量 | `"env": ["API_KEY"]` |
| `requires.config` | 必需的配置项 | `"config": ["browser.enabled"]` |
| `primaryEnv` | 主要环境变量名 | `"primaryEnv": "GEMINI_API_KEY"` |

**门控示例**：

```markdown
```

```
# 示例1：仅macOS可用
metadata: {
  "openclaw": {
    "os": ["darwin"]
  }
}

# 示例2：需要特定工具
metadata: {
  "openclaw": {
    "requires": {
      "bins": ["ffmpeg", "imagemagick"]
    }
  }
}

# 示例3：需要API 密钥
metadata: {
  "openclaw": {
    "requires": {
      "env": ["OPENAI_API_KEY"]
    },
    "primaryEnv": "OPENAI_API_KEY"
  }
}

# 示例4：需要配置启用
metadata: {
  "openclaw": {
    "requires": {
      "config": ["features.experimental"]
    }
  }
}
```text
### 8.3.4 Skills 安装器配置

**什么是安装器**：

安装器定义了如何安装Skills所需的依赖（二进制文件、包等）。

**支持的安装器类型**：
```
```

1. **Homebrew**（macOS/Linux）
2. **Node包管理器**（npm/pnpm/yarn/bun）
3. **Go**
4. **UV**（Python）
5. **Download**（直接下载）

安装器示例：

```markdown
---
name: gemini
metadata: {
  "openclaw": {
    "install": [
      {
        "id": "brew",
        "kind": "brew",
        "formula": "gemini-cli",
        "bins": ["gemini"],
        "label": "Install Gemini CLI (brew)",
        "os": ["darwin", "linux"]
      },
      {
        "id": "npm",
        "kind": "node",
        "package": "gemini-cli",
        "bins": ["gemini"],
        "label": "Install Gemini CLI (npm)",
        "global": true
      }
    ]
  }
}
---
```text
**安装器字段说明**：

| 字段 | 说明 |
|------|------|
| `id` | 安装器唯一标识 |
| `kind` | 安装器类型（brew/node/go/uv/download） |
| `formula` | Homebrew formula名称 |
| `package` | npm包名称 |

```
`bins`	安装后的二进制文件列表
`label`	显示给用户的标签
`os`	支持的操作系统
`global`	是否全局安装（Node）

Download安装器：

```markdown
metadata: {
  "openclaw": {
    "install": [
      {
        "id": "download-mac",
        "kind": "download",
        "url": "https://example.com/tool-mac.tar.gz",
        "archive": "tar.gz",
        "extract": true,
        "stripComponents": 1,
        "targetDir": "~/.openclaw/tools/my-tool",
        "bins": ["my-tool"],
        "label": "Download for macOS",
        "os": ["darwin"]
      }
    ]
  }
}
```text
**安装器选择逻辑**：
```

1. 如果列出多个安装器，Gateway会选择首选选项：
   - 优先选择brew（如果可用）
   - 其次选择node
   - 最后选择其他
2. 如果所有安装器都是download类型：
   - OpenClaw会列出每个条目
   - 让用户选择适合的构件
3. Node安装遵循配置：
   - skills.install.nodeManager（默认npm）

- 选项：npm/pnpm/yarn/bun

```
### 8.3.5 项目结构

**标准Skills目录结构**：
```

```
my-skill/
├── SKILL.md # Skills定义（必需）
├── README.md # 说明文档
├── tools/ # 工具脚本
│   ├── setup.sh # 安装脚本
│   └── cleanup.sh # 清理脚本
├── examples/ # 示例
│   └── example.md
└── assets/ # 资源文件
    └── icon.png
```

```
**SKILL.md引用文件**：

在SKILL.md中可以使用`{baseDir}`引用Skills文件夹路径：

```markdown
---
name: my-skill
description: My custom skill
---

# My Skill

## 安装

运行安装脚本：
```bash
bash {baseDir}/tools/setup.sh
```text
## 示例

查看示例：{baseDir}/examples/example.md
```

由于内容较长，让我先提交当前进度，然后继续完成剩余章节。
```

<function_calls>

git add .

实战案例1：开发1个简单的Skills

```
// src/index.ts
import { Skill, SkillConfig } from '@openclaw/sdk';

export default class HelloSkill extends Skill {
  config: SkillConfig = {
    name: 'hello-skill',
    version: '1.0.0',
    description: '1个简单的问候Skills',
    author: '你的名字',
    keywords: ['问候', '示例'],
    permissions: []
  };

  async onMessage(message: string) {
    // 检测是否包含"你好"
    if (message.includes('你好')) {
      return '你好！我是Hello Skill，很高兴认识你！😊';
    }

    // 检测是否包含"再见"
    if (message.includes('再见')) {
      return '再见！期待下次见面！👋';
    }

    return null; // 不处理其他消息
  }
}
```text
**测试Skills**：

```bash
# 启动开发服务器
npm run dev

# 在OpenClaw中测试
你：你好

OpenClaw：你好！我是Hello Skill，很高兴认识你！😊
```text
**实战案例2：封装第三方API**

```typescript
```

```
// src/index.ts
import { Skill, SkillConfig } from '@openclaw/sdk';
import axios from 'axios';

export default class WeatherSkill extends Skill {
  config: SkillConfig = {
    name: 'weather-skill',
    version: '1.0.0',
    description: '天气查询Skills',
    author: '你的名字',
    keywords: ['天气', 'API'],
    permissions: ['network']
  };

  private apiKey = 'your-api-key';
  private apiUrl = 'https://api.weather.com';

  async onMessage(message: string) {
    // 检测是否是天气查询
    const match = message.match(/(.+)的天气/);
    if (!match) return null;

    const city = match[1];

    try {
      // 调用天气API
      const response = await axios.get(`${this.apiUrl}/weather`, {
        params: {
          city: city,
          key: this.apiKey
        }
      });

      const weather = response.data;

      return `${city}的天气：
🌡️ 温度：${weather.temp}°C
☁️ 天气：${weather.condition}
💨 风力：${weather.wind}
💧 湿度：${weather.humidity}%`;

    } catch (error) {
      return `抱歉，查询${city}的天气失败了。`;
```

```
    }
  }
}
```text
**实战案例3：复杂Skills开发**

```typescript
// src/index.ts
import { Skill, SkillConfig } from '@openclaw/sdk';
import { FileManager } from './utils/file';
import { NotionAPI } from './utils/notion';

export default class NoteSkill extends Skill {
  config: SkillConfig = {
    name: 'note-skill',
    version: '1.0.0',
    description: '笔记管理Skills',
    author: '你的名字',
    keywords: ['笔记', 'Notion'],
    permissions: ['file:read', 'file:write', 'network']
  };

  private fileManager: FileManager;
  private notionAPI: NotionAPI;

  async onInit() {
    // 初始化文件管理器
    this.fileManager = new FileManager();

    // 初始化Notion API
    this.notionAPI = new NotionAPI({
      token: process.env.NOTION_TOKEN
    });
  }

  async onMessage(message: string) {
    // 保存笔记
    if (message.startsWith('保存笔记：')) {
      const content = message.replace('保存笔记：', '');
      return await this.saveNote(content);
    }

    // 搜索笔记
```

```
    if (message.startsWith('搜索笔记：')) {
      const keyword = message.replace('搜索笔记：', '');
      return await this.searchNote(keyword);
    }

    // 同步到Notion
    if (message === '同步到Notion') {
      return await this.syncToNotion();
    }

    return null;
  }

  private async saveNote(content: string) {
    try {
      // 保存到本地
      await this.fileManager.save('notes.md', content);
      return '笔记已保存到本地 ✅';
    } catch (error) {
      return '保存失败 ❌';
    }
  }

  private async searchNote(keyword: string) {
    try {
      // 搜索本地笔记
      const results = await this.fileManager.search(keyword);
      return `找到${results.length}条笔记：\n${results.join('\n')}`;
    } catch (error) {
      return '搜索失败 ❌';
    }
  }

  private async syncToNotion() {
    try {
      // 读取本地笔记
      const notes = await this.fileManager.readAll();

      // 同步到Notion
      for (const note of notes) {
        await this.notionAPI.createPage({
          title: note.title,
          content: note.content
```

```
        });
      }

      return `已同步${notes.length}条笔记到Notion ✅`;
    } catch (error) {
      return '同步失败 ❌';
    }
  }
}
```text
### 8.3.4 调试和测试技巧

**调试方法**：

**1. 使用console.log**
```typescript
async onMessage(message: string) {
  console.log('收到消息：', message);

  // 你的逻辑
  const result = await this.process(message);
  console.log('处理结果：', result);

  return result;
}
```text
**2. 使用调试器**
```bash
# 启动调试模式
npm run dev:debug

# 在VSCode中设置断点
# 按F5开始调试
```text
**3. 单元测试**
```typescript
// tests/index.test.ts
import { describe, it, expect } from 'vitest';
import HelloSkill from '../src/index';

describe('HelloSkill', () => {
  it('should respond to hello', async () => {
    const skill = new HelloSkill();
```

```
    const response = await skill.onMessage('你好');
    expect(response).toContain('你好');
  });

  it('should respond to goodbye', async () => {
    const skill = new HelloSkill();
    const response = await skill.onMessage('再见');
    expect(response).toContain('再见');
  });
});
```text
**运行测试**：
```bash
# 运行所有测试
npm test

# 运行单个测试
npm test -- index.test.ts

# 查看测试覆盖率
npm run test:coverage
```text
### 8.3.5 发布到ClawHub

**发布前检查**：

```bash
# 1. 运行测试
npm test

# 2. 检查代码规范
npm run lint

# 3. 构建生产版本
npm run build

# 4. 检查包大小
npm run size
```text
**发布步骤**：

```bash
# 1. 登录ClawHub
```

```
openclaw login

# 2. 发布Skills
openclaw skill publish

# 输出示例：
正在发布 hello-skill...
- 检查配置... ✅
- 运行测试... ✅
- 构建代码... ✅
- 上传文件... ✅
- 生成文档... ✅

发布成功！✅

Skill信息：
- 名称：hello-skill
- 版本：v1.0.0
- 链接：https://clawhub.com/skills/hello-skill

现在其他用户可以安装你的Skill了！
```text
**发布后维护**：

```bash
# 更新Skills
openclaw skill update

# 查看下载统计
openclaw skill stats

# 查看用户反馈
openclaw skill feedback
```text
---

## 8.4 Skills管理技巧

### 8.4.1 配置覆盖（~/.openclaw/openclaw.json）

**配置Skills**：

内置/托管Skills可以通过配置文件进行切换和配置：
```
```

```
```json
{
  "skills": {
    "entries": {
      "nano-banana-pro": {
        "enabled": true,
        "apiKey": "GEMINI_KEY_HERE",
        "env": {
          "GEMINI_API_KEY": "GEMINI_KEY_HERE"
        },
        "config": {
          "endpoint": "https://example.invalid",
          "model": "nano-pro"
        }
      },
      "peekaboo": {
        "enabled": true
      },
      "sag": {
        "enabled": false
      }
    }
  }
}
```text
**配置规则**：
```

1. 配置键默认匹配Skills名称
2. 如果Skills名称包含连字符，用引号括起键名
3. 如果Skills定义了metadata.openclaw.skillKey，使用该键

示例：

- Skills名称：file-search
- 配置键："file-search"（需要引号）

```
**配置字段说明**：

| 字段 | 说明 | 示例 |
|------|------|------|
| `enabled` | 启用/禁用Skills | `true` / `false` |
| `apiKey` | API 密钥（便捷字段） | `"your-api-key"` |
| `env` | 环境变量 | `{"KEY": "value"}` |
| `config` | 自定义配置 | `{"model": "gpt-4"}` |

**环境变量注入规则**：
```

1. env中的变量仅在进程中尚未设置时注入
2. apiKey为声明primaryEnv的Skills提供便捷配置
3. config用于自定义单Skills字段
4. 环境变量在智能体运行时注入，运行结束后恢复

示例配置：

```json
{
  "skills": {
    "entries": {
      // 配置Gemini Skills
      "gemini": {
        "enabled": true,
        "apiKey": "your-gemini-key",
        "config": {
          "model": "gemini-pro",
          "temperature": 0.7
        }
      },

      // 配置Banana绘图
      "nano-banana-pro": {
        "enabled": true,
        "env": {
          "GEMINI_API_KEY": "your-key",
          "BANANA_ENDPOINT": "https://api.banana.dev"
        }
      },

      // 禁用某个Skills
      "unwanted-skill": {
        "enabled": false
      }
    },

    // 仅允许特定内置Skills（可选）
    "allowBundled": [
      "file-search",
      "calendar-sync",
      "web-clipper"
    ]
  }
}
```text
**allowBundled白名单**：

```
```json
{
  "skills": {
    "allowBundled": [
      "file-search",
      "note-sync"
    ]
  }
}
```text
说明：
- 如果设置了`allowBundled`，只有列表中的内置Skills才有资格
- 托管/工作区Skills不受影响
- 用于限制内置Skills的加载

### 8.4.2 Skills监视器（自动刷新）

**什么是Skills监视器**：

Skills监视器会监视Skills文件夹，当SKILL.md文件更改时自动更新Skills快照。

**配置监视器**：

```json
{
  "skills": {
    "load": {
      "watch": true,              // 启用监视器（默认true）
      "watchDebounceMs": 250      // 防抖延迟（毫秒）
    }
  }
}
```text
**工作原理**：
```

1. OpenClaw在会话开始时对有资格的Skills进行快照
2. 监视器监视Skills文件夹的变化
3. 当SKILL.md更改时，刷新Skills列表
4. 刷新后的列表在下1个智能体轮次生效

```
**使用场景**：
```

✅ 开发Skills时实时测试
✅ 修改Skills 配置后立即生效
✅ 添加新Skills后自动加载

```
### 8.4.3 安全注意事项

**⚠️ 重要安全提醒**：
```

1. 将第三方Skills视为不受信任的代码
2. 启用前请仔细阅读Skills内容
3. 对于不受信任的输入和高风险工具，使用沙箱隔离
4. 保护API 密钥，不要泄露到日志中

安全最佳实践：

1. 审查Skills代码

```bash
# 安装前查看Skills内容
cat ~/.openclaw/skills/my-skill/SKILL.md

# 检查Skills权限要求
grep "requires" ~/.openclaw/skills/my-skill/SKILL.md
```text
**2. 使用沙箱隔离**

```json
{
  "agents": {
    "defaults": {
      "sandbox": {
        "enabled": true,
        "docker": {
          "image": "openclaw/sandbox:latest"
        }
      }
    }
  }
}
```text
**3. 保护API 密钥**

```json
{
  "skills": {
    "entries": {
      "my-skill": {
        "apiKey": "your-secret-key",  // ✅ 通过配置文件
        "env": {
          "API_KEY": "your-key"       // ✅ 通过环境变量
        }
      }
    }
  }
}
```

```text
❌ **不要**：
```markdown
---
name: my-skill
description: My skill with API key: sk-xxx7890  # ❌ 不要在SKILL.md中暴露密钥
---
```text
**4. 限制Skills权限**

```json
{
  "skills": {
    "entries": {
      "untrusted-skill": {
        "enabled": true,
        "config": {
          "allowedPaths": ["/safe/path"],  // 限制文件访问
          "allowedCommands": ["ls", "cat"]  // 限制命令执行
        }
      }
    }
  }
}
```text
**5. 定期审计**

```bash
# 列出所有已安装的Skills
openclaw skill list

# 检查Skills 配置
cat ~/.openclaw/openclaw.json | grep -A 10 "skills"

# 查看Skills日志
openclaw logs --skill my-skill
```text
### 8.4.4 性能优化

**Token影响（Skills列表）**：

当Skills有资格时，OpenClaw将可用Skills的紧凑XML列表注入到系统提示词中。
```

```
**成本计算**：
```

基础开销（仅当≥1个Skills时）：195字符

每个Skills：97字符 + XML转义的字段长度

- name（转义后）
- description（转义后）
- location（转义后）

公式：
total = 195 + Σ (97 + len(name) + len(description) + len(location))

Token估算（OpenAI风格）：
~4字符/token，所以每个Skills ≈ 24 token + 字段长度

```
**优化建议**：
```

1. 保持Skills描述简洁
   ❌ 差：This is a very detailed and comprehensive skill that can do many things including...
   ✅ 好：Generate images via Gemini API

2. 使用简短的Skills名称
   ❌ 差：super-advanced-image-generation-tool
   ✅ 好：nano-banana-pro

3. 禁用不需要的Skills
   {
   "skills": {
   "entries": {
   "unused-skill": { "enabled": false }
   }
   }
   }

4. 使用allowBundled限制内置Skills
   {
   "skills": {
   "allowBundled": ["file-search", "calendar-sync"]
   }
   }

```
**性能监控**：

```bash
# 查看Skills加载时间
openclaw skill perf

# 输出示例：
Skills加载统计：
- 总Skills数：15
- 加载时间：120ms
- Token消耗：约450 tokens
- 内存占用：2.3MB
```text
### 8.4.5 多智能体Skills管理

**单智能体 vs 共享Skills**：
```

单智能体Skills：

- 位置：/skills
- 作用域：仅该智能体可用
- 用途：项目特定功能

共享Skills：

- 位置：~/.openclaw/skills
- 作用域：所有智能体可见
- 用途：通用功能

多智能体配置示例：

```json
{
  "agents": {
    "list": [
      {
        "id": "main",
        "workspace": "/home/user/main-agent",
        "skills": {
          "entries": {
            "file-search": { "enabled": true }
          }
        }
      },
      {
        "id": "assistant",
        "workspace": "/home/user/assistant-agent",
        "skills": {
          "entries": {
            "calendar-sync": { "enabled": true }
          }
        }
      }
    ]
  }
}
```text
**共享Skills文件夹**：

```json
{
  "skills": {
    "load": {
      "extraDirs": [
        "/shared/team-skills",      // 团队共享
        "/shared/company-skills"    // 公司共享
      ]
    }
  }
}
```text

```
8.4.6 远程macOS节点（Linux Gateway）

跨平台Skills支持：

如果Gateway运行在Linux上但连接了macOS节点，OpenClaw可以将仅限macOS的Skills视为有资格。

配置示例：

```json
{
  "nodes": {
    "macos-node": {
      "host": "192.168.1.100",
      "platform": "darwin",
      "allowSystemRun": true
    }
  }
}
```text
**工作原理**：
```

1. Linux Gateway检测到macOS节点
2. 检查节点上的二进制文件
3. 将macOS专用Skills标记为可用
4. 通过nodes工具执行Skills

```
**注意事项**：
```

⚠️ 如果macOS节点离线：

- Skills仍然可见
- 调用可能失败
- 直到节点重新连接

8.4.7 Skills备份和恢复

备份Skills：

```bash
# 备份所有Skills
openclaw skill backup

# 备份到指定位置
openclaw skill backup --output ~/backups/skills.zip

# 输出示例：
正在备份Skills...
- file-search ✅
- note-sync ✅
- calendar-sync ✅

备份完成！
文件：~/backups/skills-2026-02-11.zip
大小：15.3MB
```text
**恢复Skills**：

```bash
# 从备份恢复
openclaw skill restore ~/backups/skills-2026-02-11.zip

# 输出示例：
正在恢复Skills...
- file-search ✅
- note-sync ✅
- calendar-sync ✅

恢复完成！
共恢复3个Skills
```text
**手动备份**：

```bash
# 备份Skills文件夹
cp -r ~/.openclaw/skills ~/backups/skills-$(date +%Y%m%d)
```

```
# 备份配置文件
cp ~/.openclaw/openclaw.json ~/backups/openclaw-$(date +%Y%m%d).json
```text
### 8.4.8 常见问题排查

**问题1：Skill无法加载**
```

症状：Skill安装成功但无法使用

排查步骤：

1. 检查Skill是否启用
   openclaw skill status file-search
2. 查看错误日志
   openclaw skill logs file-search
3. 检查门控条件
   cat ~/.openclaw/skills/file-search/SKILL.md | grep "requires"
4. 重新安装
   openclaw skill reinstall file-search

```
**问题2：Skill响应慢**
```

症状：Skill执行时间过长

排查步骤：

1. 查看性能报告
   openclaw skill perf file-search
2. 检查网络连接
   ping api.example.com
3. 清理缓存
   openclaw skill cache clear file-search
4. 检查依赖
   which uv python

```
**问题3：Skill配置错误**
```

症状：Skill功能异常
```

排查步骤：

1. 检查配置文件
   openclaw skill config show file-search
2. 验证环境变量
   echo $GEMINI_API_KEY
3. 重置配置
   openclaw skill config reset file-search
4. 查看文档
   openclaw skill docs file-search

```
**问题4：Skills冲突**
```

症状：多个Skills响应同一命令

排查步骤：

1. 检测冲突
   openclaw skill check
2. 查看优先级
   openclaw skill priority
3. 禁用冲突的Skills
   openclaw config set skills.entries.conflicting-skill.enabled false
4. 修改命令关键词
   编辑SKILL.md，修改触发词

📝 本章小结

本章学习了OpenClaw的Skills扩展功能：

核心内容

1. **ClawHub技能市场**
   - ClawHub介绍
   - 技能分类和搜索
   - 技能评价体系
   - Skills 安装和管理

2. **必装Skills推荐**
   - 文件管理类（3个）
   - 知识管理类（3个）
   - 日程管理类（2个）
   - 自动化类（2个）
   - 工具类（2个）

3. **自定义Skills开发**
   - Skills开发基础
   - 开发环境搭建
   - Skills结构详解
   - API接口设计
   - 调试和测试
   - 发布到ClawHub

4. **Skills管理技巧**
   - 安装和卸载
   - 配置管理
   - 冲突解决
   - 性能优化
   - 备份和恢复
   - 问题排查

实战技巧

- ✅ 选择高质量Skills
- ✅ 合理配置Skills
- ✅ 定期更新Skills

- ✅ 监控Skills性能
- ✅ 及时备份配置

下一步

- 学习第9章：多平台集成
- 掌握飞书、企微、钉钉、QQ接入
- 构建多平台AI助手

🎯 实战练习

练习1：安装必备Skills
1. 安装file-search、calendar-sync、web-clipper
2. 配置Skills参数
3. 测试Skills功能

练习2：开发简单Skills
1. 创建1个问候Skills
2. 实现基本功能
3. 测试和调试

练习3：Skills性能优化
1. 监控Skills性能
2. 找出性能瓶颈
3. 优化和改进

💡 常见问题

Q1：如何选择合适的Skills？
A：看评分、下载量、更新频率和文档质量。

Q2：Skills冲突怎么办？
A：使用`openclaw skill check`检测并自动修复。

Q3：如何开发自己的Skills？
A：参考官方文档和示例代码，从简单开始。

Q4：Skills性能差怎么办？
A：查看性能报告，优化代码，清理缓存。

```
**Q5：Skills数据如何备份？**
A：使用`openclaw skill backup`定期备份。

---

**下一章预告**：第9章将学习多平台集成，包括飞书、企业微信、钉钉、QQ Bot配置等内容。飞书作为最现代化的平台，支持WebSocket长连接、流式输出等高级功能。


---

## 8.8 百度千帆Skills生态

### 8.8.1 为什么选择百度千帆Skills？

**核心优势**：
```

✅ 官方支持：百度官方维护，稳定可靠
✅ 中文优化：针对中文场景深度优化
✅ 免费使用：大部分Skills免费
✅ 易于集成：一键安装，无需复杂配置
✅ 生态完善：覆盖搜索、学术、办公等场景

与其他Skills对比：

特性	百度千帆Skills	国际Skills	社区Skills
中文支持	⭐⭐⭐⭐⭐	⭐⭐⭐	⭐⭐⭐⭐
稳定性	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐
免费程度	⭐⭐⭐⭐⭐	⭐⭐	⭐⭐⭐⭐
文档质量	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐
更新频率	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐

8.8.2 百度搜索系列Skills

1. 百度搜索（Baidu Search）

功能介绍：
- 实时网络搜索
- 智能结果筛选
- 自动摘要生成
- 支持多种搜索类型

安装方法：
```bash
# 云端部署自带，无需安装
# 本地部署需要安装
clawhub install baidu-search
```text
**适用场景**：
- 实时信息查询
- 行业动态追踪
- 竞品分析
- 内容创作素材收集

#### 2. 百度百科（Baidu Baike）

**功能介绍**：
- 权威知识查询
- 概念解释
- 历史背景
- 相关词条推荐

**适用场景**：
- 学习新概念

- 写作背景调研
- 知识科普
- 术语解释

#### 3. 百度学术（Baidu Scholar）

**功能介绍**：
- 学术论文搜索
- 引用分析
- 作者查询
- 期刊信息

**适用场景**：
- 文献调研
- 论文阅读
- 学术研究
- 毕业论文写作

### 8.8.3 百度办公系列Skills

#### 4. 百度智能PPT（Baidu Smart PPT）

**功能介绍**：
- 自动生成PPT
- 智能排版
- 配图推荐
- 模板选择

**适用场景**：
- 产品介绍
- 工作汇报
- 培训教材
- 商业提案

### 8.8.4 获取百度API Key

**步骤1：注册百度智能云账号**
1. 访问：https://cloud.baidu.com
2. 点击"注册"
3. 填写手机号和验证码
4. 完成实名认证

**步骤2：开通千帆大模型服务**

1. 进入控制台
2. 搜索"千帆大模型"
3. 点击"立即开通"
4. 同意服务协议

**步骤3：获取API Key**
1. 进入"访问管理"
2. 点击"安全认证"
3. 选择"Access Key"
4. 点击"创建Access Key"
5. 保存Access Key ID和Secret Access Key

![百度智能云控制台 - 获取API Key和Secret Key]
(https://upload.maynor1024.live/file/1770782712874_image_17.jpg)

### 8.8.5 百度千帆MCP广场

**什么是MCP广场？**

MCP（Model Context Protocol）广场是百度千帆推出的应用连接平台，提供丰富的MCP Server，让OpenClaw可以连接各种应用和服务。

**访问地址**：https://cloud.baidu.com/product/qianfan_mcp.html

**支持的应用**：
- 办公协作：飞书、钉钉、企业微信
- 知识管理：Notion、语雀、印象笔记
- 开发工具：GitHub、GitLab、Jenkins
- 数据分析：Tableau、Power BI

### 8.8.6 效率提升数据

**使用百度Skills前后对比**：

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---|---|---|---|---|
| 信息搜索 | 30分钟 | 2分钟 | 28分钟 | 93.3% |
| 概念查询 | 15分钟 | 1分钟 | 14分钟 | 93.3% |
| 论文检索 | 60分钟 | 5分钟 | 55分钟 | 91.7% |
| PPT制作 | 120分钟 | 10分钟 | 110分钟 | 91.7% |
| **平均** | **225分钟** | **18分钟** | **207分钟** | **92%** |

---

## 相关资源

- 百度千帆官网：https://cloud.baidu.com/product/qianfan.html
- 百度千帆Skills文档：https://cloud.baidu.com/doc/qianfan/s/Mmlda41a2
- 百度千帆MCP广场：https://cloud.baidu.com/product/qianfan_mcp.html
- API Key获取：https://console.bce.baidu.com/iam/#/iam/apikey/list

---

## 8.9 API服务Skills

> 💡 **本节目标**：学会将第三方API服务封装为Skills，实现AI绘图、数据同步、视频生成、语音合成等功能。

### 8.9.1 为什么需要API服务Skills

**使用场景**：
- 🎨 **AI绘图**：Banana、Midjourney、DALL-E
- 📝 **数据同步**：Notion、Airtable、语雀
- 🎬 **视频生成**：SeeDream、可灵（Kling）
- 🔊 **语音合成**：海螺TTS、MiniMax Music
- 🌐 **Web服务**：各类API接口

**优势**：
- 统一管理：所有API都是Skills
- 易于使用：一条命令安装
- 自动更新：跟随Skills更新
- 社区共享：开发者贡献

### 8.9.2 AI绘图Skills：Banana

#### 什么是Banana

**Banana介绍**：
- Nano Banana Pro是一款AI绘图工具
- 支持文生图、图生图、图片编辑
- 质量高、速度快、价格便宜

**为什么选择Banana**：
- ✅ 质量优秀：媲美Midjourney
- ✅ 速度快：10秒出图

- ✅ 价格低：$0.02/张
- ✅ 易集成：API简单易用

#### 安装和配置

```bash
# 安装Banana Skills
openclaw skill install banana-draw

# 配置API Key
openclaw config set banana.apiKey "your-api-key"

# 测试
你：用Banana画一只可爱的猫
OpenClaw：正在生成图片...
[图片]
```text
#### 实战案例

**案例1：手机上画图**

```text
场景：在飞书上用OpenClaw画图

步骤：
1. 打开飞书
2. 找到OpenClaw
3. 发送：画1个赛博朋克风格的城市
4. 等待10秒
5. 收到图片

优势：
- 无需打开电脑
- 随时随地画图
- 自动保存到相册
```text
**案例2：批量生成图片**

```text
你：帮我生成3张图片：
1. 科技感的办公室
2. 未来城市
3. AI机器人
```

OpenClaw：正在生成...
[图片1]
[图片2]
[图片3]

全部完成 ✅
```text
#### 提示词优化技巧

**基本原则**：
1. 描述清晰具体
2. 包含风格关键词
3. 指定画面元素
4. 控制画面比例

**优秀提示词示例**：

❌ 差：画一只猫
✅ 好：一只可爱的橘猫，坐在窗台上，阳光洒在身上，温暖的色调，高清摄影，景深效果

❌ 差：画个城市
✅ 好：赛博朋克风格的未来城市，霓虹灯闪烁，高楼林立，夜景，雨后街道，电影级画质，广角镜头

**风格关键词**：
- 写实风格：photorealistic, 4K, detailed
- 卡通风格：cartoon, cute, colorful
- 赛博朋克：cyberpunk, neon, futuristic
- 水彩画：watercolor, soft, artistic
- 油画：oil painting, classical, textured

### 8.9.3 数据同步Skills：Notion

#### Notion介绍

**Notion的优势**：
- ✅ 功能强大：笔记、数据库、项目管理
- ✅ 跨平台：Web、Mac、Windows、iOS、Android
- ✅ 协作友好：团队共享、权限管理
- ✅ API完善：易于集成

#### 安装和配置
```

```bash
# 安装Notion Skills
openclaw skill install notion-sync

# 配置Token
openclaw config set notion.token "your-notion-token"

# 配置Database ID
openclaw config set notion.databaseId "your-database-id"

# 测试
你：把这段内容保存到Notion
OpenClaw：已保存到Notion ✅
```text
#### 实战案例

**案例1：笔记同步**

```text
你：保存这段笔记到Notion：
[笔记内容]

OpenClaw：已保存 ✅
标题：[自动生成]
标签：#笔记
链接：https://notion.so/...
```text
**案例2：任务管理**

```text
你：添加任务：完成项目报告
截止日期：明天
优先级：高

OpenClaw：任务已添加 ✅
```text
**案例3：知识库构建**

```text
你：把这篇文章保存到知识库
[文章链接]

OpenClaw：正在处理...
```

- 提取内容 ✅
- 生成摘要 ✅
- 添加标签 ✅
- 保存到Notion ✅

完成！
```text
### 8.9.4 视频生成Skills

#### 视频生成工具介绍

**SeeDream**：
- 文生视频
- 图生视频
- 视频编辑
- 价格：$0.1/秒

**可灵（Kling）**：
- 国产视频生成
- 质量优秀
- 价格实惠
- 支持中文

#### 安装和配置

```bash
# 安装视频生成Skills
openclaw skill install video-gen

# 配置SeeDream
openclaw config set seedream.apiKey "your-api-key"

# 配置可灵
openclaw config set kling.apiKey "your-api-key"
```text
#### 实战案例

```text
你：生成1个5秒的视频：
海浪拍打沙滩，日落时分

OpenClaw：正在生成...
预计时间：2分钟
```
```

```
[视频生成完成]
时长：5秒
分辨率：1080p
大小：15MB
```text
### 8.9.5 语音合成Skills

#### 语音合成工具

**海螺TTS**：
- 多种音色
- 自然流畅
- 支持中英文
- 价格：$0.01/千字

**MiniMax Music**：
- AI音乐生成
- 多种风格
- 高质量输出

#### 安装和配置

```bash
# 安装TTS Skills
openclaw skill install tts

# 配置海螺TTS
openclaw config set heluo.apiKey "your-api-key"

# 测试
你：把这段文字转成语音：
[文字内容]

OpenClaw：正在生成...
[语音文件]
```text
### 8.9.6 自定义API封装

如果ClawHub上没有你需要的API Skills，可以自己封装：

```typescript
// src/custom-api-skill.ts
```

```
import { Skill } from '@openclaw/sdk';
import axios from 'axios';

export default class CustomAPISkill extends Skill {
  private apiKey = process.env.CUSTOM_API_KEY;
  private apiUrl = 'https://api.example.com/v1';

  async onMessage(message: string) {
    // 检测API请求
    const match = message.match(/调用API:(.+)/);
    if (!match) return null;

    const params = match[1];

    try {
      // 调用API
      const response = await axios.post(
        `${this.apiUrl}/endpoint`,
        { params },
        {
          headers: {
            'Authorization': `Bearer ${this.apiKey}`
          }
        }
      );

      return `API调用成功：${response.data}`;

    } catch (error) {
      return '调用失败，请稍后重试';
    }
  }
}
```text
### 8.9.7 成本对比

**AI绘图成本对比**：

| 工具 | 价格 | 质量 | 速度 |
|------|------|------|------|
| Banana | $0.02 | ⭐⭐⭐⭐ | 10秒 |
| Midjourney | $0.08 | ⭐⭐⭐⭐⭐ | 30秒 |
| DALL-E | $0.04 | ⭐⭐⭐⭐ | 15秒 |
```
```

| Stable Diffusion | 免费 | ⭐⭐⭐ | 20秒 |

语音合成成本对比：

工具	价格	质量	语言支持
海螺TTS	$0.01/千字	⭐⭐⭐⭐	中英文
Azure TTS	$0.016/千字	⭐⭐⭐⭐⭐	多语言
Google TTS	$0.016/千字	⭐⭐⭐⭐	多语言

8.9.8 最佳实践

1. API Key管理
```bash
# 使用环境变量
export BANANA_API_KEY="your-key"
export NOTION_TOKEN="your-token"

# 或使用配置文件
openclaw config set api.keys '{
  "banana": "your-key",
  "notion": "your-token"
}'
```text
**2. 错误处理**
```typescript
try {
  const result = await callAPI();
  return result;
} catch (error) {
  if (error.code === 'RATE_LIMIT') {
    return '请求过于频繁，请稍后重试';
  }
  return '调用失败，请检查配置';
}
```text
**3. 成本控制**
- 设置每日调用上限
- 使用缓存减少重复调用
- 选择性价比高的服务
- 监控API使用量

**4. 安全建议**

- 不要在代码中硬编码API Key
- 定期更换API Key
- 使用环境变量或配置文件
- 限制API权限范围

---

## 8.10 常用工具安装案例

> 💡 **实战指南**：OpenClaw的Skills系统依赖一些外部工具，本节介绍两个常用工具的完整安装过程。

### 8.10.1 1Password CLI安装（密码管理）

**工具简介**：
1Password CLI (op) 是1Password的命令行工具，让OpenClaw能够安全地访问和管理你的密码库中的凭证、API 密钥等敏感信息。

**使用场景**：
- 安全存储API 密钥
- 自动填充登录信息
- 团队密码共享
- CI/CD流程中的密钥管理

**安装步骤**：

```bash
# macOS安装（推荐使用Homebrew）
brew install --cask 1password-cli

# 验证安装
which op
# 输出：/usr/local/bin/op

op --version
# 输出：2.32.1
```text
**配置使用**：

1. **启用桌面应用集成**：
   - 打开1Password桌面应用
   - 进入设置 → 开发者
   - 启用"与1Password CLI集成"

```
2. **登录账户**：
```bash
# 登录1Password账户
op signin

# 或指定账户域名
op signin my.1password.com
```text
3. **在OpenClaw中使用**：
```bash
# 获取密码
op item get "GitHub" --fields password

# 获取API 密钥
op item get "OpenAI API" --fields credential

# 列出所有项目
op item list
```text
**OpenClaw集成示例**：
```

你：从1Password获取GitHub的API 密钥

OpenClaw：正在获取...
✅ 已获取GitHub API 密钥：ghp_xxxxxxxxxxxx

你：用这个密钥创建一个新的GitHub仓库

OpenClaw：好的，正在创建...
✅ 仓库创建成功！

安全提示：
- ⚠️ 不要在代码中硬编码密码
- ✅ 使用1Password CLI安全管理所有凭证
- ✅ 定期更换API 密钥
- ✅ 为不同项目使用不同的凭证

8.10.2 camsnap安装（摄像头捕获）

工具简介：
camsnap是一个用于捕获RTSP/ONVIF摄像头画面的命令行工具，让OpenClaw能够获取网络摄像头的实时画面或录制视频片段。

使用场景：
- 智能家居监控
- 安防系统集成
- 自动化截图和录像
- 视频分析和处理

系统要求：
- macOS（Apple Silicon或Intel）
- Homebrew包管理器
- 网络摄像头（支持RTSP/ONVIF协议）

安装步骤：

步骤1：检查系统架构

```bash
# 检查CPU架构
uname -m
# 输出：arm64（Apple Silicon）或 x86_64（Intel）
```text
**步骤2：确保使用ARM64版本的Homebrew（Apple Silicon Mac）**

如果你的Mac是Apple Silicon（M1/M2/M3芯片），需要确保安装了ARM64版本的Homebrew：

```bash
# 检查当前brew位置
which brew
# 应该输出：/opt/homebrew/bin/brew（ARM64版本）
```

```
# 如果输出：/usr/local/bin/brew（Intel版本），需要安装ARM64版本

# 安装ARM64版本的Homebrew
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# 配置shell环境（zsh）
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zshrc
eval "$(/opt/homebrew/bin/brew shellenv)"

# 验证brew版本
brew --version
which brew
# 应该输出：/opt/homebrew/bin/brew
```text
**步骤3：安装camsnap**

```bash
# 添加tap源
brew tap steipete/tap

# 安装camsnap
brew install camsnap

# 验证安装
which camsnap
# 输出：/opt/homebrew/bin/camsnap

camsnap --version
# 输出：0.2.0
```text
**步骤4：创建符号链接（可选，解决兼容性问题）**

如果OpenClaw无法找到camsnap，可以创建符号链接：

```bash
# 创建符号链接到/usr/local/bin
sudo ln -sf /opt/homebrew/bin/camsnap /usr/local/bin/camsnap

# 验证符号链接
ls -la /usr/local/bin/camsnap
# 输出：lrwxr-xr-x@ 1 root  wheel  25 Feb 13 14:47 /usr/local/bin/camsnap ->
/opt/homebrew/bin/camsnap
```

```
# 测试
/usr/local/bin/camsnap --version
# 输出：0.2.0
```text
**使用示例**：

```bash
# 捕获单帧画面
camsnap --url rtsp://192.168.1.100:554/stream --output snapshot.jpg

# 录制视频片段（10秒）
camsnap --url rtsp://192.168.1.100:554/stream --duration 10 --output video.mp4

# 指定分辨率
camsnap --url rtsp://192.168.1.100:554/stream --width 1920 --height 1080 --output
hd.jpg
```text
**OpenClaw集成示例**：
```

你：拍一张客厅摄像头的照片

OpenClaw：正在捕获画面...
✅ 已保存到：~/Pictures/living-room-2026-02-13-14-30.jpg
[发送图片]

你：录制10秒的门口监控视频

OpenClaw：正在录制...
⏱️ 录制中... 10秒
✅ 已保存到：~/Videos/door-monitor-2026-02-13-14-31.mp4
```

```
**故障排查**：

**问题1：安装失败，提示"arm64 architecture is required"**

```bash
# 原因：使用了Intel版本的brew
# 解决：安装ARM64版本的Homebrew（见步骤2）

# 检查当前brew架构
brew config | grep -E "CPU|Homebrew"
# 应该显示：CPU: octa-core 64-bit arm_*
```text
**问题2：OpenClaw提示"Missing: bin:camsnap"**

```bash
# 原因：OpenClaw无法找到camsnap
# 解决：创建符号链接（见步骤4）

# 或者重启OpenClaw Gateway
openclaw gateway restart
```text
**问题3：摄像头连接失败**

```bash
# 检查摄像头URL是否正确
# RTSP URL格式：rtsp://用户名:密码@IP地址:端口/路径

# 测试摄像头连接
camsnap --url rtsp://admin:password@192.168.1.100:554/stream --test

# 查看详细错误信息
camsnap --url rtsp://192.168.1.100:554/stream --verbose
```
```

安全提示：

- ⚠️ 不要在公网暴露摄像头RTSP端口
- ✅ 使用强密码保护摄像头
- ✅ 定期更新摄像头固件
- ✅ 限制OpenClaw对摄像头的访问权限

8.10.3 工具安装最佳实践

通用安装流程：

1. **检查系统要求**

```
# 检查操作系统
uname -s

# 检查CPU架构
uname -m

# 检查可用空间
df -h
```

2. **选择合适的安装方式**

   - Homebrew（推荐，适合macOS/Linux）
   - npm/yarn（适合Node.js工具）
   - pip/uv（适合Python工具）
   - 直接下载二进制文件

3. **验证安装**

```
# 检查工具是否在PATH中
which <tool-name>

# 检查版本
<tool-name> --version

# 测试基本功能
<tool-name> --help
```

4. **配置环境变量**

```
# 添加到~/.zshrc或~/.bashrc
export TOOL_API_KEY="your-key"
export TOOL_CONFIG_PATH="~/.config/tool"

# 重新加载配置
source ~/.zshrc
```

5. **集成到OpenClaw**

```
# 重启OpenClaw Gateway
openclaw gateway restart

# 验证工具可用性
openclaw tools list
```

常见问题排查：

问题	原因	解决方案
找不到命令	未添加到PATH	检查安装路径，添加到PATH
权限不足	需要管理员权限	使用sudo或管理员身份运行
依赖缺失	缺少必需的库	安装依赖包
版本不兼容	工具版本过旧	更新到最新版本
架构不匹配	ARM/Intel不兼容	安装对应架构的版本

推荐工具列表：

工具	用途	安装命令
1Password CLI	密码管理	`brew install --cask 1password-cli`
camsnap	摄像头捕获	`brew install steipete/tap/camsnap`
ffmpeg	视频处理	`brew install ffmpeg`
imagemagick	图片处理	`brew install imagemagick`
jq	JSON处理	`brew install jq`
gh	GitHub CLI	`brew install gh`
uv	Python包管理	`brew install uv`

📝 本章小结

通过本章学习，你已经掌握：

1. **Skills生态**：了解1715+个技能的庞大生态
2. **ClawHub市场**：学会搜索、安装、管理Skills
3. **核心Skills**：掌握必装的5个核心技能

4. **安装方法**：三种安装方式灵活选择
5. **实战应用**：三大场景的完整应用
6. **安全使用**：权限管理和安全防护
7. **Skills开发**：从零开发自己的Skills
8. **管理技巧**：更新、备份、故障排查
9. **API服务**：封装第三方API为Skills
10. **工具安装**：1Password CLI和camsnap完整安装流程

🎯 实战练习

1. 安装5个核心Skills并测试
2. 尝试开发1个简单的Skills
3. 封装一个你常用的API服务
4. 构建一个完整的自动化工作流

💡 进阶建议

1. 探索ClawHub上的更多Skills
2. 参与Skills社区贡献
3. 开发并分享你的Skills
4. 优化Skills性能和体验

下一章预告：第9章将学习多平台集成，包括飞书、企业微信、钉钉、QQ等平台的Bot配置，实现随时随地使用OpenClaw。飞书作为最现代化的平台将优先介绍。

返回目录：README

第9章 多平台集成

> 💡 **本章目标**：学会将OpenClaw接入飞书、企业微信、钉钉、QQ等国内主流平台，实现随时随地使用AI助手。飞书作为最现代化、开发友好的平台，将优先介绍。

📱 本章内容

- 9.1 飞书Bot配置
  - 9.1.1 飞书机器人介绍
  - 9.1.2 快速开始
  - 9.1.3 创建飞书应用
  - 9.1.4 配置 OpenClaw
  - 9.1.5 启动并测试
  - 9.1.6 访问控制
  - 9.1.7 群组配置
  - 9.1.8 获取群组/用户 ID
  - 9.1.9 高级配置
  - 9.1.10 多账号配置
  - 9.1.11 多 Agent 配置
    - 9.1.11.1 配合飞书使用
    - 9.1.11.2 实战案例：4个专业助手
    - 9.1.11.3 配置注意事项
    - 9.1.11.4 故障排查
    - 9.1.11.5 配置对比
    - 9.1.11.6 使用建议
    - 9.1.11.7 本地多 Agent 管理（无需绑定 IM 平台）⭐新增
- 9.2 企业微信Bot配置
- 9.3 钉钉Bot配置
- 9.4 QQ Bot配置

9.1 飞书Bot配置

> 💡 **状态**：生产就绪，支持机器人私聊和群组，使用 WebSocket 长连接模式接收消息。

9.1.1 飞书机器人介绍

企业微信的优势：

1. **工作场景**
   - 与微信互通
   - 企业级功能
   - 安全可靠
2. **使用便捷**

- 手机随时访问
- 消息即时送达
- 支持文件传输

3. **功能丰富**
   - 支持文本、图片、文件
   - 支持群聊
   - 支持应用集成
4. **免费使用**
   - 基础功能免费
   - 无用户数限制
   - 稳定可靠

适用场景：

- ✅ 个人效率提升
- ✅ 团队协作
- ✅ 客户服务
- ✅ 工作自动化

9.2.2 企业微信注册

注册步骤：

步骤1：访问企业微信官网

访问 https://work.weixin.qq.com/
```text
**步骤2：注册企业**

```text
1. 点击"立即注册"
2. 选择"企业注册"
3. 填写企业信息：
   - 企业名称
   - 行业类型
   - 企业规模
   - 管理员信息
4. 验证手机号
5. 完成注册
```text
**步骤3：完善企业信息**

```text
1. 登录管理后台
2. 完善企业资料
3. 添加成员（可选）
4. 设置部门（可选）
```text
### 9.2.3 创建企业微信应用

**步骤1：进入应用管理**

```text
1. 登录企业微信管理后台
2. 点击"应用管理"
3. 点击"创建应用"
```text
**步骤2：填写应用信息**

```text
- 应用名称：OpenClaw助手
- 应用Logo：上传Logo图片
- 应用简介：AI智能助手
- 可见范围：选择可使用的成员
```text
**步骤3：获取应用凭证**

```text
创建成功后，记录以下信息：
- AgentId：应用ID
- Secret：应用密钥
- CorpId：企业ID
```yaml
### 9.2.4 配置 OpenClaw

> 💡 **前置要求**：请先完成 OpenClaw 的基础安装和配置，详见 [第2章：环境搭建](../01-basics/02-
```

installation.md)。

云端部署（推荐）

如果使用云端部署，参考腾讯云官方教程可一键完成企微接入：
https://cloud.tencent.com/developer/article/2625147

本地配置

```bash
# 安装企微Skills
openclaw skill install wework-bot

# 配置应用信息
openclaw config set wework.corpId "your-corp-id"
openclaw config set wework.agentId "your-agent-id"
openclaw config set wework.secret "your-secret"

# 启动企微Bot
openclaw wework start
```text
### 9.2.5 实战案例

**案例1：个人助手Bot**

```text
场景：在企业微信上使用 OpenClaw作为个人助手

功能：
- 文件搜索
- 日程管理
- 知识管理
- 任务提醒

使用示例：
你：帮我找一下上个月的发票
OpenClaw：找到3张发票，已发送给你

你：明天下午3点提醒我开会
OpenClaw：已设置提醒 ✅
```text
**案例2：团队协作Bot**

```text
场景：团队群聊中使用 OpenClaw

功能：
- 会议记录
- 任务分配
- 进度追踪
- 文档管理
```

```
使用示例：
成员A：@OpenClaw 记录今天的会议要点
OpenClaw：好的，会议记录如下...

成员B：@OpenClaw 查询项目进度
OpenClaw：项目进度：已完成60%...
```text
**案例3：客户服务Bot**

```text
场景：客户咨询自动回复

功能：
- 常见问题解答
- 产品介绍
- 订单查询
- 售后服务

使用示例：
客户：你们的产品有哪些功能？
OpenClaw：产品主要功能包括...

客户：如何购买？
OpenClaw：购买方式有以下几种...
```text
### 9.2.6 手机端使用技巧

**技巧1：快速访问**
```text
1. 打开企业微信
2. 点击"工作台"
3. 找到"OpenClaw助手"
4. 开始对话
```text
**技巧2：语音输入**
```text
1. 点击语音按钮
2. 说出你的需求
3. OpenClaw自动识别并处理
```text
**技巧3：文件传输**
```text
1. 点击"+"号
2. 选择"文件"
3. 发送给OpenClaw
4. OpenClaw自动处理
```text
### 9.2.7 常见问题解决

**问题1：应用无法接收消息**
```text
原因：回调URL配置错误
```

```
解决方案：
1. 检查回调URL是否正确
2. 确保服务器可访问
3. 检查Token和EncodingAESKey
```text
**问题2：消息发送失败**
```text
原因：权限不足或配置错误

解决方案：
1. 检查应用权限
2. 确认Secret正确
3. 查看错误日志
```text
**问题3：域名备案问题**
```text
提示：域名备案主体与企业微信认证主体不一致

解决方案：
参考官方教程：
https://cloud.tencent.com/developer/article/2626187
```text
---

## 9.3 钉钉Bot配置

### 9.3.1 钉钉机器人介绍

**钉钉的优势**：

1. **企业办公**
   - 考勤打卡
   - 审批流程
   - 日程管理

2. **即时通讯**
   - 消息必达
   - 已读回执
   - 群聊功能

3. **开放平台**
   - API丰富
   - 文档完善
   - 社区活跃

4. **免费使用**
   - 基础功能免费
   - 稳定可靠

### 9.3.2 创建钉钉应用
```
```

步骤1：注册钉钉开放平台

```text
https://open.dingtalk.com/
```text
**步骤2：创建企业内部应用**

```text
1. 登录开放平台
2. 点击"应用开发"
3. 选择"企业内部开发"
4. 点击"创建应用"
```text
**步骤3：配置应用信息**

```text
应用名称：OpenClaw助手
应用描述：AI智能助手
应用图标：上传图标
开发方式：企业内部开发
```text
**步骤4：获取凭证**

```text
记录以下信息：
- AppKey：应用Key
- AppSecret：应用密钥
- AgentId：应用ID
```text
### 9.3.3 配置 OpenClaw

> 💡 **前置要求**：请先完成 OpenClaw 的基础安装和配置，详见 [第2章：环境搭建](../01-basics/02-installation.md)。

**云端部署（推荐）**

参考腾讯云官方教程可一键完成钉钉接入：

```text
https://cloud.tencent.com/developer/article/2626553
```text
**本地配置**

```bash
# 安装钉钉Skills并配置
openclaw skill install dingtalk-bot
openclaw config set dingtalk.appKey "your-app-key"
openclaw config set dingtalk.appSecret "your-app-secret"
openclaw config set dingtalk.agentId "your-agent-id"
openclaw dingtalk start
```text
### 9.3.4 实战案例

**案例1：工作助手**
```text
功能：
- 考勤提醒
- 日程管理
- 任务追踪
- 报表生成

使用示例：
你：今天的日程安排
OpenClaw：今天有3个会议...

你：生成本周工作报表
OpenClaw：报表已生成 ✅
```text
**案例2：审批流程**
```text
功能：
- 审批提醒
- 流程查询
- 自动填单
- 进度追踪

使用示例：
你：我的待审批事项
OpenClaw：有2个待审批...

你：帮我填写请假单
OpenClaw：请假单已填写 ✅
```text
---

## 9.1 飞书Bot配置

> 💡 **状态**：生产就绪，支持机器人私聊和群组，使用 WebSocket 长连接模式接收消息。

### 9.1.1 飞书机器人介绍

**飞书的优势**：

1. **现代化办公**
   - 文档协作
   - 多维表格
   - 视频会议

2. **高效沟通**
   - 消息卡片
   - 互动组件
   - 流式输出

3. **开发友好**
   - API设计优秀

```
    - 文档详细
    - WebSocket长连接

4. **免费使用**
    - 功能强大
    - 稳定可靠

9.1.2 快速开始

添加飞书渠道有两种方式：

方式一：通过安装向导添加（推荐）

如果您刚安装完 OpenClaw，可以直接运行向导：

```bash
openclaw setup
```text
向导会引导您完成：
1. 创建飞书应用并获取凭证
2. 配置应用凭证
3. 启动网关

✅ **完成配置后**，您可以使用以下命令检查网关状态：
```bash
openclaw gateway status      # 查看网关运行状态
openclaw logs --follow       # 查看实时日志
```text
**方式二：通过命令行添加**

如果您已经完成了初始安装，可以用以下命令添加飞书渠道：

```bash
openclaw channels add
```text
然后根据交互式提示选择 Feishu，输入 App ID 和 App Secret 即可。

✅ **完成配置后**，您可以使用以下命令管理网关：
```bash
openclaw gateway status      # 查看网关运行状态
openclaw gateway restart     # 重启网关以应用新配置
openclaw logs --follow       # 查看实时日志
```text
### 9.1.3 第一步：创建飞书应用

#### 1. 打开飞书开放平台

访问 [飞书开放平台](https://open.feishu.cn/app)，使用飞书账号登录。

> 💡 **Lark（国际版）**：请使用 https://open.larksuite.com/app，并在配置中设置 `domain: "lark"`。

#### 2. 创建应用
```

1. 点击 **创建企业自建应用**
2. 填写应用名称和描述
3. 选择应用图标

![飞书开放平台 - 创建企业自建应用]
(https://upload.maynor1024.live/file/1770734336224_image_1770734318.jpg)

3. 获取应用凭证

在应用的 **凭证与基础信息** 页面，复制：
- **App ID**（格式如 `cli_xxx`）
- **App Secret**

❗ **重要**：请妥善保管 App Secret，不要分享给他人。

![飞书应用凭证 - App ID和App Secret]
(https://upload.maynor1024.live/file/1770734332380_image_1770734319.jpg)

4. 配置应用权限

在 **权限管理** 页面，点击 **批量导入** 按钮，粘贴以下 JSON 配置一键导入所需权限：

```json
{
  "scopes": {
    "tenant": [
      "aily:file:read",
      "aily:file:write",
      "application:application.app_message_stats.overview:readonly",
      "application:application:self_manage",
      "application:bot.menu:write",
      "cardkit:card:write",
      "contact:user.employee_id:readonly",
      "corehr:file:download",
      "docs:document.content:read",
      "event:ip_list",
      "im:chat",
      "im:chat.access_event.bot_p2p_chat:read",
      "im:chat.members:bot_access",
      "im:message",
      "im:message.group_at_msg:readonly",
      "im:message.group_msg",
      "im:message.p2p_msg:readonly",
      "im:message:readonly",
      "im:message:send_as_bot",
      "im:resource",
      "sheets:spreadsheet",
      "wiki:wiki:readonly"
    ],
    "user": [
      "aily:file:read",
```

```
      "aily:file:write",
      "im:chat.access_event.bot_p2p_chat:read"
    ]
  }
}
```text
![飞书应用权限配置 - 批量导入JSON权限]
(https://upload.maynor1024.live/file/1770734343156_image_1770734320.jpg)

#### 5. 启用机器人能力

在 **应用能力** > **机器人** 页面：
1. 开启机器人能力
2. 配置机器人名称

![飞书机器人配置 - 启用机器人功能]
(https://upload.maynor1024.live/file/1770734349201_image_1770734321.jpg)

#### 6. 配置事件订阅

⚠️ **重要提醒**：在配置事件订阅前，请务必确保已完成以下步骤：
1. 运行 `openclaw channels add` 添加了 Feishu 渠道
2. 网关处于启动状态（可通过 `openclaw gateway status` 检查状态）

在 **事件订阅** 页面：

**步骤1：选择长连接模式**
1. 选择 **使用长连接接收事件**（WebSocket 模式）

**步骤2：添加事件**
2. 添加事件：`im.message.receive_v1`（接收消息）

**步骤3：配置必需权限（重要）**

在配置事件订阅的同时，请确保在 **权限管理** 页面已添加以下权限：

| 权限标识 | 权限名称 | 是否必需 | 说明 |
|---------|---------|---------|------|
| `im:message` | 获取与发送单聊、群组消息 | ✅ 必需 | 接收和发送消息 |
| `im:message:send_as_bot` | 以应用身份发消息 | ✅ 必需 | 以机器人身份回复 |
| `contact:contact.base:readonly` | 获取通讯录基本信息 | ✅ 必需 | 识别用户身份 |

> 💡 **为什么需要 `contact:contact.base:readonly` 权限？**
>
> 这个权限用于获取用户的基本信息（如用户名、部门等），OpenClaw需要这些信息来：
> - ✅ 识别消息发送者
> - ✅ 实现访问控制（allowlist/denylist）
> - ✅ 提供个性化服务
> - ✅ 记录对话历史
>
> ⚠️ **如果缺少此权限，机器人将无法正常响应消息！**
```
```

配置截图示例：

![飞书权限配置 - 通讯录权限](https://upload.maynor1024.live/file/1771065454975_image-20260214183727712.png)

⚠️ **注意**：如果网关未启动或渠道未添加，长连接设置将保存失败。

![飞书事件订阅 - 使用长连接接收消息](https://upload.maynor1024.live/file/1770734352151_image_1770734322.jpg)

常见错误排查：

如果遇到 "Gateway start blocked: set gateway.mode=local" 错误：
```bash
# 确保配置文件中设置了 gateway.mode
{
  "gateway": {
    "mode": "local"
  }
}
```text
如果遇到 "Gateway auth is set to token, but no token is configured" 错误：
```bash
# 方式1：在配置文件中设置 token
{
  "gateway": {
    "auth": {
      "mode": "token",
      "token": "your-secure-token"
    }
  }
}

# 方式2：使用环境变量
export OPENCLAW_GATEWAY_TOKEN="your-secure-token"
```text
#### 7. 发布应用

1. 在 **版本管理与发布** 页面创建版本
2. 提交审核并发布
3. 等待管理员审批（企业自建应用通常自动通过）

### 9.1.4 第二步：配置 OpenClaw

#### 安装 Feishu 插件

```bash
# 安装 Feishu 插件
openclaw plugins install @openclaw/feishu

# 本地 checkout（在 git 仓库内运行）
openclaw plugins install ./extensions/feishu
```

```text
#### 通过向导配置（推荐）

运行以下命令，根据提示粘贴 App ID 和 App Secret：

```bash
openclaw channels add
```text
选择 **Feishu**，然后输入您在第一步获取的凭证即可。

#### 通过配置文件配置

编辑 `~/.openclaw/openclaw.json`：

```json
{
  "channels": {
    "feishu": {
      "enabled": true,
      "dmPolicy": "pairing",
      "accounts": {
        "main": {
          "appId": "cli_xxx",
          "appSecret": "xxx",
          "botName": "我的AI助手"
        }
      }
    }
  }
}
```text
#### 通过环境变量配置

```bash
export FEISHU_APP_ID="cli_xxx"
export FEISHU_APP_SECRET="xxx"
```text
#### Lark（国际版）域名配置

如果您的租户在 Lark（国际版），请设置域名为 `lark`：

```json
{
  "channels": {
    "feishu": {
      "domain": "lark",
      "accounts": {
        "main": {
          "appId": "cli_xxx",
          "appSecret": "xxx"
        }
      }
```
```

```
    }
  }
}
```text
### 9.1.5 第三步：启动并测试

#### 1. 启动网关

```bash
# 安装并启动网关
openclaw gateway install

# 检查网关状态
openclaw gateway status

# 查看实时日志
openclaw logs --follow
```text
**网关启动成功的标志：**
```

✅ Gateway: running (pid xxxxx, state active)
✅ Gateway target: ws://127.0.0.1:18789
✅ Source: local loopback

```
#### 2. 发送测试消息

在飞书中找到您创建的机器人，发送一条消息，例如："hi"。

**在日志中应该能看到：**
```

HEARTBEAT_OK
hi
connected | running
agent main | session main (heartbeat) | local-antigravity/gemini-3-pro-high
```

3. 配对授权

默认情况下（`dmPolicy: "pairing"`），机器人会回复一个 **配对码**。您需要批准此代码：

```bash
# 查看待审批的配对请求
openclaw pairing list feishu

# 批准配对（替换 <配对码> 为实际收到的代码）
openclaw pairing approve feishu <配对码>

# 示例
openclaw pairing approve feishu ABC123
```text
批准后即可正常对话。

**如果不想使用配对模式：**
```json
{
  "channels": {
    "feishu": {
      "dmPolicy": "open",
      "allowFrom": ["*"]
    }
  }
}
```text
### 9.1.6 访问控制

#### 私聊访问

**默认策略**：`dmPolicy: "pairing"`，陌生用户会收到配对码

**批准配对**：
```bash
openclaw pairing list feishu           # 查看待审批列表
openclaw pairing approve feishu <CODE> # 批准
```text
**白名单模式**：通过 `channels.feishu.allowFrom` 配置允许的用户 Open ID

#### 群组访问

**1. 群组策略**（`channels.feishu.groupPolicy`）：
- `"open"` = 允许群组中所有人（默认）
- `"allowlist"` = 仅允许 `groupAllowFrom` 中的用户
- `"disabled"` = 禁用群组消息

**2. @提及要求**（`channels.feishu.groups.<chat_id>.requireMention`）：
- `true` = 需要 @机器人才响应（默认）
- `false` = 无需 @也响应

### 9.1.7 群组配置示例

#### 允许所有群组，需要 @提及（默认行为）

```json
{
  "channels": {
    "feishu": {
      "groupPolicy": "open"
      // 默认 requireMention: true
    }
  }
}
```text
#### 允许所有群组，无需 @提及

需要为特定群组配置：

```json
{
  "channels": {
    "feishu": {
      "groups": {
        "oc_xxx": { "requireMention": false }
      }
    }
  }
}
```text
#### 仅允许特定用户在群组中使用

```json
{
  "channels": {
    "feishu": {
      "groupPolicy": "allowlist",
      "groupAllowFrom": ["ou_xxx", "ou_yyy"]
    }
  }
}
```text
### 9.1.8 获取群组/用户 ID

#### 获取群组 ID（chat_id）

群组 ID 格式为 `oc_xxx`，可以通过以下方式获取：

**方法一**（推荐）：
1. 启动网关并在群组中 @机器人发消息
2. 运行 `openclaw logs --follow` 查看日志中的 `chat_id`

**方法二**：使用飞书 API 调试工具获取机器人所在群组列表。

#### 获取用户 ID（open_id）

用户 ID 格式为 `ou_xxx`，可以通过以下方式获取：

**方法一**（推荐）：
1. 启动网关并给机器人发消息
2. 运行 `openclaw logs --follow` 查看日志中的 `open_id`

**方法二**：查看配对请求列表，其中包含用户的 Open ID：
```bash
openclaw pairing list feishu
```text
### 9.1.9 高级配置

### 自定义菜单

添加常用命令在菜单上

![飞书机器人菜单 - 添加常用命令快捷入口](https://upload.maynor1024.live/file/1770874980945_image-20260212134245771.png)

这里我新建了三个常用命令：新建对话，列出技能，继续。

![自定义菜单示例 - 新建对话/列出技能/继续](https://upload.maynor1024.live/file/1770874990637_image-20260212134300933.png)

#### 多账号配置

OpenClaw 支持同时管理多个飞书机器人，这在以下场景非常有用：
- 不同团队使用不同的机器人
- 测试环境和生产环境分离
- 不同功能的专用机器人
- 主备机器人配置

**基础配置示例（2个机器人）：**

```json
{
  "channels": {
    "feishu": {
      "enabled": true,
      "dmPolicy": "pairing",
      "accounts": {
        "bot1": {
          "appId": "cli_xxxxxxxxxxxxxxxx",
```

```
          "appSecret": "your-app-secret-1",
          "botName": "OpenClaw助手1",
          "enabled": true
        },
        "bot2": {
          "appId": "cli_yyyyyyyyyyyyyyyy",
          "appSecret": "your-app-secret-2",
          "botName": "OpenClaw助手2",
          "enabled": true
        }
      },
      "domain": "feishu",
      "groupPolicy": "open",
      "connectionMode": "websocket",
      "requireMention": true,
      "renderMode": "auto",
      "streaming": true,
      "blockStreaming": true,
      "replyToMode": "all"
    }
  },
  "gateway": {
    "port": 18789,
    "mode": "local",
    "bind": "lan",
    "auth": {
      "mode": "token",
      "token": "your-secure-token-here"
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "your-provider/your-model"
      },
      "workspace": "/path/to/your/workspace",
      "compaction": {
        "mode": "safeguard"
      },
      "maxConcurrent": 4,
      "subagents": {
        "maxConcurrent": 8
      }
    }
  }
}
```text
> 💡 **实战提示**：上面的配置示例来自真实的多机器人部署案例。注意 `appSecret` 和 `token` 在生产环境中应该妥善保管，不要提交到代码仓库。

**多机器人配置示例（4个专业助手）：**
```
```

```json
{
  "channels": {
    "feishu": {
      "enabled": true,
      "dmPolicy": "pairing",
      "accounts": {
        "main-assistant": {
          "appId": "cli_main_xxxxxx",
          "appSecret": "your-main-secret",
          "botName": "主助理",
          "enabled": true
        },
        "content-creator": {
          "appId": "cli_content_xxxxxx",
          "appSecret": "your-content-secret",
          "botName": "内容创作助手",
          "enabled": true
        },
        "tech-dev": {
          "appId": "cli_tech_xxxxxx",
          "appSecret": "your-tech-secret",
          "botName": "技术开发助手",
          "enabled": true
        },
        "ai-news": {
          "appId": "cli_news_xxxxxx",
          "appSecret": "your-news-secret",
          "botName": "AI资讯助手",
          "enabled": true
        }
      },
      "domain": "feishu",
      "groupPolicy": "open",
      "connectionMode": "websocket",
      "requireMention": true,
      "streaming": true,
      "blockStreaming": true,
      "replyToMode": "all"
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "anthropic/claude-sonnet-4"
      },
      "workspace": "/path/to/workspace",
      "compaction": {
        "mode": "safeguard"
      },
      "maxConcurrent": 4,
      "subagents": {
```

```
        "maxConcurrent": 8
      }
    }
  }
}
```text
> ⚠️ **重要提示**：在多账号配置中，不需要使用 `bindings` 来绑定不同的 agent。所有机器人会自动共享 `agents.defaults` 配置。如果需要不同的模型，可以在对话中使用 `/model` 命令切换。

**配置说明：**

| 参数 | 说明 | 必填 |
|------|------|------|
| `accounts.<id>` | 账号唯一标识符（自定义） | ✅ |
| `appId` | 飞书应用的 App ID | ✅ |
| `appSecret` | 飞书应用的 App Secret | ✅ |
| `botName` | 机器人显示名称 | ❌ |
| `enabled` | 是否启用该账号 | ❌ (默认 true) |

**多机器人使用场景：**

1. **一人公司/独立开发者**
   - 主助理：任务分发、日程管理
   - 内容创作助手：文章、视频脚本
   - 技术开发助手：代码开发、调试
   - AI资讯助手：行业动态追踪

2. **团队协作**
   - 技术团队助手：代码审查、技术讨论
   - 产品团队助手：需求分析、用户反馈
   - 运营团队助手：数据分析、内容运营
   - 测试助手：测试环境专用

3. **环境分离**
   - 生产环境助手：正式业务使用
   - 测试环境助手：功能测试
   - 开发环境助手：开发调试

重要

在 **事件订阅** 页面：

**步骤1：选择长连接模式**
1. 选择 **使用长连接接收事件**（WebSocket 模式）

**步骤2：添加事件**
2. 添加事件：`im.message.receive_v1`（接收消息）

**步骤3：配置必需权限**
3. 在 **权限管理** 页面，确保已添加：
   - ✅ `im:message`（获取与发送单聊、群组消息）
   - ✅ `im:message:send_as_bot`（以应用身份发消息）
```
```

- ✅ `contact:contact.base:readonly`（获取通讯录基本信息）⭐ 必需

> 💡 缺少 `contact:contact.base:readonly` 权限会导致机器人无法识别用户，无法正常响应消息。
2. 添加事件：`im.message.receive_v1`（接收消息）

![飞书长连接配置 - WebSocket模式事件订阅](https://upload.maynor1024.live/file/1770880072503_image-20260212150740769.png)

![飞书事件订阅成功 - 长连接已建立](https://upload.maynor1024.live/file/1770880042274_image-20260212150708631.png)

实战场景1：团队分离

为不同团队创建专用机器人：

```json
{
  "channels": {
    "feishu": {
      "accounts": {
        "tech-team": {
          "appId": "cli_tech_xxx",
          "appSecret": "tech_secret",
          "botName": "技术团队助手",
          "enabled": true
        },
        "sales-team": {
          "appId": "cli_sales_xxx",
          "appSecret": "sales_secret",
          "botName": "销售团队助手",
          "enabled": true
        },
        "hr-team": {
          "appId": "cli_hr_xxx",
          "appSecret": "hr_secret",
          "botName": "HR助手",
          "enabled": true
        }
      }
    }
  }
}
```text
**实战场景2：环境分离**

测试环境和生产环境使用不同的机器人：

```json
```

```
{
  "channels": {
    "feishu": {
      "accounts": {
        "production": {
          "appId": "cli_prod_xxx",
          "appSecret": "prod_secret",
          "botName": "OpenClaw生产环境",
          "enabled": true,
          "dmPolicy": "pairing"
        },
        "staging": {
          "appId": "cli_staging_xxx",
          "appSecret": "staging_secret",
          "botName": "OpenClaw测试环境",
          "enabled": true,
          "dmPolicy": "open"
        },
        "development": {
          "appId": "cli_dev_xxx",
          "appSecret": "dev_secret",
          "botName": "OpenClaw开发环境",
          "enabled": false
        }
      }
    }
  }
}
```text
**实战场景3：功能分离**

不同功能使用专用机器人：

```json
{
  "channels": {
    "feishu": {
      "accounts": {
        "general": {
          "appId": "cli_general_xxx",
          "appSecret": "general_secret",
          "botName": "通用助手",
          "enabled": true
        },
        "code-review": {
          "appId": "cli_code_xxx",
          "appSecret": "code_secret",
          "botName": "代码审查助手",
          "enabled": true
        },
        "document": {
          "appId": "cli_doc_xxx",
```

```
          "appSecret": "doc_secret",
          "botName": "文档助手",
          "enabled": true
        }
      }
    }
  }
}
```text
**配合多 Agent 使用**

将不同的飞书机器人绑定到不同的 Agent，实现更精细的功能分离：

```json
{
  "agents": {
    "list": [
      {
        "id": "general-agent",
        "workspace": "/home/user/general",
        "agentDir": "/home/user/.openclaw/agents/general/agent"
      },
      {
        "id": "code-agent",
        "workspace": "/home/user/code-review",
        "agentDir": "/home/user/.openclaw/agents/code/agent"
      },
      {
        "id": "doc-agent",
        "workspace": "/home/user/document",
        "agentDir": "/home/user/.openclaw/agents/doc/agent"
      }
    ]
  },
  "channels": {
    "feishu": {
      "accounts": {
        "general": {
          "appId": "cli_general_xxx",
          "appSecret": "general_secret",
          "botName": "通用助手"
        },
        "code-review": {
          "appId": "cli_code_xxx",
          "appSecret": "code_secret",
          "botName": "代码审查助手"
        },
        "document": {
          "appId": "cli_doc_xxx",
          "appSecret": "doc_secret",
          "botName": "文档助手"
        }
```

```
      }
    }
  },
  "bindings": [
    {
      "agentId": "general-agent",
      "match": {
        "channel": "feishu",
        "account": "general"
      }
    },
    {
      "agentId": "code-agent",
      "match": {
        "channel": "feishu",
        "account": "code-review"
      }
    },
    {
      "agentId": "doc-agent",
      "match": {
        "channel": "feishu",
        "account": "document"
      }
    }
  ]
}
```text
**管理多个机器人**

```bash
# 查看所有飞书账号状态
openclaw channels list feishu

# 启用特定账号
openclaw channels enable feishu backup

# 禁用特定账号
openclaw channels disable feishu test

# 重启特定账号
openclaw channels restart feishu main

# 查看特定账号的日志
openclaw logs --channel feishu --account main --follow
```text
**配置文件位置**

```bash
# 主配置文件
~/.openclaw/openclaw.json
```

```
# 或者使用独立的渠道配置文件
~/.openclaw/channels/feishu.json
```text
**独立配置文件示例：**

```bash
# 创建独立配置文件
mkdir -p ~/.openclaw/channels
nano ~/.openclaw/channels/feishu.json
```text
```json
{
  "enabled": true,
  "accounts": {
    "main": {
      "appId": "cli_xxx",
      "appSecret": "xxx",
      "botName": "主机器人"
    },
    "backup": {
      "appId": "cli_yyy",
      "appSecret": "yyy",
      "botName": "备用机器人"
    }
  }
}
```

注意事项：

1. **App ID 和 App Secret 必须唯一**
   - 每个机器人必须使用不同的飞书应用
   - 不能多个账号共用同一个 App ID
2. **账号标识符命名规范**
   - 使用小写字母和连字符
   - 避免使用特殊字符
   - 建议使用有意义的名称（如 `tech-team`、`production`）
3. **启用/禁用控制**
   - `enabled: true` - 账号启用，机器人会接收和处理消息
   - `enabled: false` - 账号禁用，机器人不会接收消息
   - 可以随时通过修改配置文件或命令行切换
4. **网关重启**
   - 修改配置后需要重启网关：`openclaw gateway restart`
   - 或者重新加载配置：`openclaw channels reload`
5. **日志查看**
   - 多账号时，日志会标注账号标识符
   - 使用 `--account` 参数过滤特定账号的日志

故障排查：

问题1：某个机器人收不到消息

```
# 检查账号是否启用
openclaw channels status feishu

# 查看该账号的日志
openclaw logs --channel feishu --account main --follow

# 检查配置是否正确
openclaw config get channels.feishu.accounts.main
```text
**问题2：多个机器人冲突**

确保每个机器人使用不同的飞书应用：
- 不同的 App ID
- 不同的 App Secret
- 在飞书开放平台创建多个应用

**问题3：切换账号不生效**

```bash
# 重启网关
openclaw gateway restart

# 或者重新加载配置
openclaw channels reload feishu
```text
**问题4：配置验证失败 - bindings 错误**
```

Error: bindings.0.match: Unrecognized key: "account"
```

```
**原因**：在多账号配置中，不需要使用 `bindings` 来绑定 agent。

**解决方案**：
1. 删除配置文件中的 `bindings` 部分
2. 所有机器人会自动使用 `agents.defaults` 配置
3. 如果需要不同模型，在对话中使用 `/model` 命令切换

**正确的配置结构**：
```json
{
  "channels": {
    "feishu": {
      "accounts": {
        "bot1": { ... },
        "bot2": { ... }
      }
    }
  },
  "agents": {
    "defaults": {
      "model": { "primary": "your-model" },
      "workspace": "/path/to/workspace"
    }
  }
  // ❌ 不需要 bindings
}
```text
**问题5：配置后运行 openclaw doctor 报错**

```bash
# 运行诊断
openclaw doctor

# 如果提示配置问题，运行自动修复
openclaw doctor --fix

# 验证配置
openclaw doctor
# 应该看到：✅ Config valid
```text
**最佳实践：**

1. **使用有意义的账号名称**
   ```json
   "accounts": {
     "prod-main": { ... },      // 生产环境主机器人
     "prod-backup": { ... },    // 生产环境备份
     "test": { ... }            // 测试环境
   }
```

2. **为不同环境使用不同的策略**
```

```
"production": {
  "dmPolicy": "pairing",     // 生产环境需要配对
  "groupPolicy": "allowlist" // 群组白名单
},
"development": {
  "dmPolicy": "open",        // 开发环境开放访问
  "groupPolicy": "open"      // 群组开放
}
```

3. **定期备份配置**

```
# 备份配置文件
cp ~/.openclaw/openclaw.json ~/.openclaw/openclaw.json.backup

# 或使用 OpenClaw 备份命令
openclaw backup create
```

4. **使用环境变量管理敏感信息**

```
# 在 ~/.bashrc 或 ~/.zshrc 中设置
export FEISHU_MAIN_APP_ID="cli_xxx"
export FEISHU_MAIN_APP_SECRET="xxx"
export FEISHU_BACKUP_APP_ID="cli_yyy"
export FEISHU_BACKUP_APP_SECRET="yyy"
```

然后在配置文件中引用：

```
{
  "channels": {
    "feishu": {
      "accounts": {
        "main": {
          "appId": "${FEISHU_MAIN_APP_ID}",
          "appSecret": "${FEISHU_MAIN_APP_SECRET}"
        }
      }
    }
  }
}
```

流式输出

飞书支持通过交互式卡片实现流式输出，机器人会实时更新卡片内容显示生成进度。

```
{
  "channels": {
    "feishu": {
      "streaming": true,      // 启用流式卡片输出（默认 true）
      "blockStreaming": true  // 启用块级流式（默认 true）
    }
  }
}
```text
如需禁用流式输出（等待完整回复后一次性发送），可设置 `streaming: false`。

#### 消息引用

在群聊中，机器人的回复可以引用用户发送的原始消息，让对话上下文更加清晰。

```json
{
  "channels": {
    "feishu": {
      "replyToMode": "all",  // 账户级别配置（默认 "all"）
      "groups": {
        "oc_xxx": {
          "replyToMode": "first"  // 特定群组可以覆盖
        }
      }
    }
  }
}
```text
`replyToMode` 值说明：
- `"off"` = 不引用原消息（私聊默认值）
- `"first"` = 仅在第一条回复时引用原消息
- `"all"` = 所有回复都引用原消息（群聊默认值）

#### 多 Agent 路由

通过 `bindings` 配置，您可以用一个飞书机器人对接多个不同功能或性格的 Agent：

```json
{
  "agents": {
    "list": [
      { "id": "main" },
      {
        "id": "clawd-fan",
        "workspace": "/home/user/clawd-fan",
        "agentDir": "/home/user/.openclaw/agents/clawd-fan/agent"
      },
      {
        "id": "clawd-xi",
        "workspace": "/home/user/clawd-xi",
```

```
        "agentDir": "/home/user/.openclaw/agents/clawd-xi/agent"
      }
    ]
  },
  "bindings": [
    {
      "agentId": "main",
      "match": {
        "channel": "feishu",
        "peer": { "kind": "dm", "id": "ou_28b31a88..." }
      }
    },
    {
      "agentId": "clawd-fan",
      "match": {
        "channel": "feishu",
        "peer": { "kind": "dm", "id": "ou_0fe6b1c9..." }
      }
    },
    {
      "agentId": "clawd-xi",
      "match": {
        "channel": "feishu",
        "peer": { "kind": "group", "id": "oc_xxx..." }
      }
    }
  ]
}
```text
### 9.1.10 常用命令

#### 机器人命令

| 命令 | 说明 |
|------|------|
| `/status` | 查看机器人状态 |
| `/reset` | 重置对话会话 |
| `/model` | 查看/切换模型 |

#### 网关管理命令

| 命令 | 说明 |
|------|------|
| `openclaw gateway status` | 查看网关运行状态 |
| `openclaw gateway install` | 安装/启动网关服务 |
| `openclaw gateway stop` | 停止网关服务 |
| `openclaw gateway restart` | 重启网关服务 |
| `openclaw logs --follow` | 实时查看日志输出 |

### 9.1.11 故障排除

#### 机器人在群组中不响应
```
```

```
1. 检查机器人是否已添加到群组
2. 检查是否 @了机器人（默认需要 @提及）
3. 检查 `groupPolicy` 是否为 `"disabled"`
4. 查看日志：`openclaw logs --follow`

#### 机器人收不到消息

**可能原因及解决方案**：

1. **检查应用是否已发布并审批通过**
   ```bash
   # 在飞书开放平台查看应用状态
   # 确保应用已通过审核并发布
```

2. **检查事件订阅是否配置正确**
   - ✅ 已选择"使用长连接接收事件"（WebSocket模式）
   - ✅ 已添加事件：`im.message.receive_v1`
   - ✅ 长连接状态显示"已连接"
3. **检查权限配置是否完整**⭐ 重要

   缺少权限会导致机器人无法正常工作，请确保已添加以下权限：

权限标识	权限名称	检查方法
`im:message`	获取与发送单聊、群组消息	在权限管理页面查看
`im:message:send_as_bot`	以应用身份发消息	在权限管理页面查看
`contact:contact.base:readonly`	获取通讯录基本信息	⭐ 必需，否则无法识别用户

   如何检查权限：

```
1. 登录飞书开放平台
2. 进入你的应用
3. 点击"权限管理"
4. 确认上述三个权限都已添加
5. 如果缺少，点击"添加权限"补充
```

   常见错误：
   - ❌ 只添加了 `im:message`，忘记添加 `contact:contact.base:readonly`
   - ❌ 权限添加后未重新发布应用
   - ❌ 权限范围设置不正确
4. **检查网关状态**
```

```
# 查看网关是否正常运行
openclaw gateway status

# 查看实时日志
openclaw logs --follow
```

5. **检查渠道配置**

```
# 查看飞书渠道配置
openclaw channels list

# 确认 appId 和 appSecret 正确
openclaw config get channels.feishu
```

6. 检查应用是否已发布并审批通过
7. 检查事件订阅是否配置正确（`im.message.receive_v1`）
8. 检查是否选择了 **长连接** 模式
9. 检查应用权限是否完整
10. 检查网关是否正在运行：`openclaw gateway status`
11. 查看实时日志：`openclaw logs --follow`

配置文件 JSON 语法错误

错误示例：

JSON5 parse error at line 443: Python True/False vs JSON true/false

```text
**解决方案：**
```bash
# 检查 JSON 语法
cat ~/.openclaw/openclaw.json | python -m json.tool

# 常见错误：
# ❌ "enabled": True   (Python 语法)
# ✅ "enabled": true   (JSON 语法)

# ❌ 多余的逗号
# ✅ 最后一项不要逗号
```text
#### 网关启动失败

**错误1：Gateway start blocked**
```bash
# 错误信息
Gateway start blocked: set gateway.mode=local (current: unset)

# 解决方案：在配置文件中添加
{
  "gateway": {
    "mode": "local"
  }
}
```text
**错误2：Gateway auth token 未配置**
```bash
# 错误信息
Gateway auth is set to token, but no token is configured

# 解决方案1：配置文件
{
  "gateway": {
    "auth": {
      "mode": "token",
      "token": "your-secure-token"
    }
  }
}

# 解决方案2：环境变量
export OPENCLAW_GATEWAY_TOKEN="your-secure-token"
```text
**错误3：插件未找到**
```bash
# 错误信息
Config validation failed: plugins.entries.qqbot: plugin not found: qqbot
```
```

```
# 解决方案：移除未安装的插件配置
{
  "plugins": {
    "entries": {
      "feishu": {
        "enabled": true
      }
      // 移除 qqbot, ddingtalk, wecom 等未安装的插件
    }
  }
}
```text
**错误4：工作空间路径错误**
```bash
# 错误信息
run error: Error: ENOENT: no such file or directory, mkdir '/root'

# 解决方案：修正 workspace 路径（macOS 示例）
{
  "agents": {
    "defaults": {
      "workspace": "/Users/yourusername/clawd"  // 使用正确的 macOS 路径
    }
  }
}
```text
#### App Secret 泄露怎么办

1. 在飞书开放平台重置 App Secret
2. 更新配置文件中的 App Secret
3. 重启网关：`openclaw gateway restart`

#### 发送消息失败

1. 检查应用是否有 `im:message:send_as_bot` 权限
2. 检查应用是否已发布
3. 查看日志获取详细错误信息：`openclaw logs --follow`

#### 网关端口被占用

```bash
# 错误信息
Port 18789 is already in use

# 解决方案1：停止现有网关
openclaw gateway stop

# 解决方案2：使用不同端口
{
  "gateway": {
    "port": 18790
  }
```

```
}
```text
#### 配置修改不生效

```bash
# 修改配置后必须重启网关
openclaw gateway restart

# 或重新加载配置
openclaw channels reload feishu

# 检查配置是否正确加载
openclaw config get channels.feishu
```text
### 9.1.12 配置参考

| 配置项 | 说明 | 默认值 |
|--------|------|--------|
| `channels.feishu.enabled` | 启用/禁用渠道 | `true` |
| `channels.feishu.domain` | API 域名（`feishu` 或 `lark`） | `feishu` |
| `channels.feishu.accounts.<id>.appId` | 应用 App ID | - |
| `channels.feishu.accounts.<id>.appSecret` | 应用 App Secret | - |
| `channels.feishu.dmPolicy` | 私聊策略 | `pairing` |
| `channels.feishu.allowFrom` | 私聊白名单（open_id 列表） | - |
| `channels.feishu.groupPolicy` | 群组策略 | `open` |
| `channels.feishu.groupAllowFrom` | 群组白名单 | - |
| `channels.feishu.groups.<chat_id>.requireMention` | 是否需要 @提及 | `true` |
| `channels.feishu.textChunkLimit` | 消息分块大小 | `2000` |
| `channels.feishu.mediaMaxMb` | 媒体大小限制 | `30` |
| `channels.feishu.streaming` | 启用流式卡片输出 | `true` |
| `channels.feishu.blockStreaming` | 启用块级流式 | `true` |

#### dmPolicy 策略说明

| 值 | 行为 |
|----|------|
| `"pairing"` | **默认**。未知用户收到配对码，管理员批准后才能对话 |
| `"allowlist"` | 仅 `allowFrom` 列表中的用户可对话，其他静默忽略 |
| `"open"` | 允许所有人对话（需在 allowFrom 中加 `"*"`） |
| `"disabled"` | 完全禁止私聊 |

### 9.1.13 支持的消息类型

#### 接收
- ✅ 文本消息
- ✅ 图片
- ✅ 文件
- ✅ 音频
- ✅ 视频
- ✅ 表情包

#### 发送
```
```

```
- ✅ 文本消息
- ✅ 图片
- ✅ 文件
- ✅ 音频
- ⚠️ 富文本（部分支持）

### 9.1.14 与飞书生态集成

**集成飞书文档**
```

功能：

- 创建文档
- 编辑文档
- 分享文档
- 权限管理

示例：
你：把这段内容保存到飞书文档
OpenClaw：已保存到飞书文档 ✅
链接：https://...

```
**集成飞书多维表格**
```

功能：

- 创建表格
- 添加数据
- 查询数据
- 数据分析

示例：
你：把发票信息添加到多维表格
OpenClaw：已添加3条记录 ✅

```
**集成飞书日历**
```

功能：

- 创建日程
- 修改日程
- 删除日程
- 日程提醒

示例：
你：明天下午3点开会
OpenClaw：已添加到飞书日历 ✅

9.1.15 实战案例：配置双机器人

> 💡 **真实案例**：本节展示一个实际的双机器人配置案例，适用于需要分离不同功能或团队的场景。

场景说明

某团队需要两个飞书机器人：
- **机器人1**：用于日常办公和通用任务
- **机器人2**：用于特定项目或测试环境

完整配置步骤

1. 在飞书开放平台创建两个应用

分别创建两个企业自建应用，获取：
- 机器人1：App ID `cli_xxxxxxxxxxxxxxxx`，App Secret
- 机器人2：App ID `cli_yyyyyyyyyyyyyyyy`，App Secret

2. 配置 OpenClaw

编辑 `~/.openclaw/openclaw.json`：

```json
{
  "meta": {
    "lastTouchedVersion": "2026.2.6-3",
    "lastTouchedAt": "2026-02-08T09:49:58.322Z"
  },
  "channels": {
    "feishu": {
      "enabled": true,
      "dmPolicy": "pairing",
      "accounts": {
        "bot1": {
          "appId": "cli_xxxxxxxxxxxxxxxx",
          "appSecret": "your-app-secret-1",
          "botName": "OpenClaw助手1",
          "enabled": true
        },
        "bot2": {
          "appId": "cli_yyyyyyyyyyyyyyyy",
          "appSecret": "your-app-secret-2",
          "botName": "OpenClaw助手2",
          "enabled": true
        }
      },
      "domain": "feishu",
      "groupPolicy": "open",
      "connectionMode": "websocket",
```

```
      "requireMention": true,
      "renderMode": "auto",
      "streaming": true,
      "blockStreaming": true,
      "replyToMode": "all"
    }
  },
  "gateway": {
    "port": 18789,
    "mode": "local",
    "bind": "lan",
    "auth": {
      "mode": "token",
      "token": "your-secure-random-token-here"
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "your-model-provider/your-model"
      },
      "workspace": "/path/to/your/workspace",
      "compaction": {
        "mode": "safeguard"
      },
      "maxConcurrent": 4
    }
  },
  "plugins": {
    "entries": {
      "feishu": {
        "enabled": true
      }
    }
  }
}
```text
**3. 配置飞书应用权限**

为两个应用分别配置权限（批量导入 JSON，参见 9.1.3 节）。

**4. 配置事件订阅**

为两个应用分别配置：
- 选择 **使用长连接接收事件**
- 添加事件：`im.message.receive_v1`

**5. 启动网关**

```bash
# 启动网关
openclaw gateway install
```

```
# 检查状态
openclaw gateway status

# 应该看到：
# ✅ Gateway: running (pid 57344, state active)
# ✅ Gateway target: ws://127.0.0.1:18789
```text
**6. 测试机器人**

在飞书中分别给两个机器人发送消息：
```

你：hi
机器人：[配对码] 请管理员批准配对
```

7. 批准配对

```bash
# 查看配对请求
openclaw pairing list feishu

# 批准机器人1
openclaw pairing approve feishu <配对码1>

# 批准机器人2
openclaw pairing approve feishu <配对码2>
```text
**8. 验证运行**

查看日志确认两个机器人都在正常运行：

```bash
openclaw logs --follow

# 应该看到：
# HEARTBEAT_OK
# hi
# connected | running
# agent main | session main (heartbeat)
```text
#### 常见问题处理

**问题1：配置文件 JSON 语法错误**

```bash
# 错误：JSON5 parse error at line 443
# 原因：使用了 Python 语法（True/False）而非 JSON 语法（true/false）

# 检查语法
cat ~/.openclaw/openclaw.json | python -m json.tool

# 修正：
# ❌ "enabled": True
# ✅ "enabled": true
```text
**问题2：网关启动失败**

```bash
# 错误：Gateway start blocked: set gateway.mode=local
# 解决：确保配置了 gateway.mode

{
  "gateway": {
    "mode": "local"
  }
}
```

```text
**问题3：工作空间路径错误**

```bash
# 错误：ENOENT: no such file or directory, mkdir '/root'
# 原因：配置文件中使用了 Linux 路径，但实际是 macOS

# 修正（macOS）：
{
  "agents": {
    "defaults": {
      "workspace": "/Users/yourusername/clawd"
    }
  }
}
```text
**问题4：插件未找到**

```bash
# 错误：plugin not found: qqbot
# 原因：配置文件中引用了未安装的插件

# 解决：只保留已安装的插件
{
  "plugins": {
    "entries": {
      "feishu": {
        "enabled": true
      }
      // 移除 qqbot, ddingtalk, wecom 等
    }
  }
}
```text
#### 配置检查清单

- [ ] 两个飞书应用已创建
- [ ] App ID 和 App Secret 已获取
- [ ] 配置文件 JSON 语法正确
- [ ] gateway.mode 已设置为 "local"
- [ ] gateway.auth.token 已配置
- [ ] workspace 路径正确（macOS/Linux）
- [ ] 只配置了已安装的插件
- [ ] 两个应用的权限已配置
- [ ] 两个应用的事件订阅已配置（长连接）
- [ ] 两个应用已发布
- [ ] 网关已启动并运行正常
- [ ] 两个机器人都已配对批准
- [ ] 日志显示正常运行

#### 成功标志
```

配置成功后，你应该看到：

```bash
# 网关状态
$ openclaw gateway status
✅ Gateway: running (pid xxxxx, state active)
✅ Gateway target: ws://127.0.0.1:18789

# 日志输出
$ openclaw logs --follow
HEARTBEAT_OK
hi
connected | running
agent main | session main (heartbeat) | your-model-provider/your-model
tokens 25k/200k (13%)
```text
两个机器人都可以正常接收和回复消息！🎉

---

## 9.4 QQ Bot配置

### 9.4.1 QQ机器人介绍

**QQ的优势**：

1. **用户基础**
   - 用户量大
   - 覆盖面广
   - 使用习惯

2. **社交属性**
   - 群聊活跃
   - 互动性强
   - 娱乐功能

3. **开放平台**
   - QQ频道
   - QQ群机器人
   - API支持

4. **免费使用**
   - 基础功能免费
   - 易于上手

### 9.4.2 创建QQ机器人

> ⚠️ **重要提示**：QQ开放平台需要先注册账号，不是直接用QQ登录！请务必先完成注册。

**步骤1：注册QQ开放平台账号**

```
1. **访问QQ开放平台**：
```

https://q.qq.com/

2. **注册新账号**：
- ⚠️ 不是QQ登录，需要单独注册
- 点击"注册"按钮
- 填写注册信息
- 完成邮箱/手机验证

3. **登录平台**：
- 使用刚注册的账号登录
- 不要使用QQ扫码登录

步骤2：创建机器人

1. **进入机器人管理**：
- 登录后点击"机器人"
- 点击"创建机器人"

2. **填写机器人信息**：
- 机器人名称：自定义（如：我的AI助手）
- 机器人头像：上传图片
- 机器人简介：简单描述功能
- 点击"创建"

3. **等待审核**：
- 提交后等待审核（通常几分钟）
- 审核通过后即可使用

步骤3：配置机器人

1. **获取机器人凭证**：
- 进入机器人详情页
- 点击"开发管理"
- 记录以下信息：
  - **机器人ID**（BotAppID）
  - **机器人密钥**（Bot Secret）

2. **配置IP白名单**：
- 在"开发管理"页面
- 找到"IP白名单"设置
- 添加你的服务器公网IP地址
- 点击"保存"

3. **添加测试用户**：
- 在"管理" → "成员管理"
- 点击"添加成员"
- 输入你的QQ号
- 将自己添加为测试用户

4. **扫码添加机器人好友**：
- 在机器人详情页找到二维码
- 用手机QQ扫码

```
- 添加机器人为好友

**步骤4：配置 OpenClaw连接**

1. **获取服务器IP地址**：
- 如果使用腾讯云，在控制台查看公网IP
- 记录这个IP地址

2. **在腾讯云Lighthouse配置**（如果使用腾讯云）：
- 登录腾讯云：https://console.cloud.tencent.com/lighthouse
- 进入实例详情
- 点击"应用管理"标签
- 找到"QQ机器人配置"区域
- 填入：
  - 机器人ID（BotAppID）
  - 机器人密钥（Bot Secret）
- 点击"应用配置"

3. **本地配置方式**：
```bash
# 运行配置向导
openclaw onboard

# 选择 QuickStart
# 选择模型（如 Kimi 2.5）
# 输入模型 API Key
# 选择通道：QQ
# 输入机器人ID和密钥
```
```

9.4.3 配置 OpenClaw

> 💡 **前置要求**：请先完成 OpenClaw 的基础安装和配置，详见 第2章：环境搭建。

方式一：使用腾讯云Lighthouse（推荐）

如果你使用腾讯云Lighthouse部署OpenClaw，配置非常简单：

1. **进入应用管理**：
   - 登录腾讯云控制台
   - 进入轻量应用服务器
   - 点击实例 → "应用管理"
2. **配置QQ机器人**：
   - 找到"QQ机器人配置"区域
   - 填入机器人ID和密钥
   - 点击"应用配置"
   - 等待配置生效
3. **验证连接**：
   - 打开手机QQ
   - 给机器人发送消息："你好"

- 如果收到回复，说明配置成功

方式二：本地配置

如果你是本地部署或其他云服务器，使用命令行配置：

```
# 1. 运行配置向导
openclaw onboard

# 2. 选择配置选项
# - 选择 Yes 接受风险
# - 选择 QuickStart（快速开始）

# 3. 配置模型
# - 选择模型供应商（如 Moonshot AI）
# - 输入 API Key
# - 选择默认模型（如 kimi-code/kimi-for-codi）

# 4. 配置通道
# - 选择通道：QQ
# - 输入机器人ID（BotAppID）
# - 输入机器人密钥（Bot Secret）

# 5. 配置Skills和Hooks
# - Skills：选择 Yes，可以先不安装
# - Hooks：选择 session-memory

# 6. 重启服务
# - 选择 Yes 重启 gateway 服务

# 7. 测试连接
# - 选择打开 TUI（终端界面）
# - 或直接在QQ中测试
```text
**方式三：手动编辑配置文件**

```bash
# 编辑配置文件
nano ~/.openclaw/config.json

# 添加QQ配置
{
  "channels": {
    "qq": {
      "enabled": true,
      "botAppId": "你的机器人ID",
      "botSecret": "你的机器人密钥",
      "profiles": ["default"]
    }
  }
}

# 重启服务
systemctl --user restart openclaw-gateway.service
```text
**启动Gateway服务**
```
```

```bash
# 方式1：前台运行（用于测试）
openclaw gateway --port 18789 --verbose

# 方式2：后台运行（推荐）
nohup openclaw gateway --port 18789 --verbose > /dev/null 2>&1 &

# 方式3：使用systemd（最稳定）
systemctl --user enable openclaw-gateway.service
systemctl --user start openclaw-gateway.service
```text
**验证配置**

```bash
# 查看服务状态
systemctl --user status openclaw-gateway.service

# 查看日志
journalctl --user -u openclaw-gateway.service -f

# 测试连接
# 在QQ中给机器人发送消息："你好"
```text
### 9.4.4 实战案例

**案例1：个人助手**
```

功能：

- 日常对话
- 信息查询
- 任务提醒
- 娱乐互动

使用示例：
你：今天天气怎么样？
OpenClaw：今天晴天，15-25°C

你：提醒我明天交作业
OpenClaw：已设置提醒 ✅

```
**案例2：群管理**
```

功能：

- 群公告
- 成员管理
- 消息统计
- 自动回复

使用示例：
管理员：@OpenClaw 发布公告
```

OpenClaw：公告已发布 ✅

成员：@OpenClaw 查询群规
OpenClaw：群规如下...

```
**案例3：娱乐互动**
```

功能：

- 聊天对话
- 讲笑话
- 猜谜语
- 玩游戏

使用示例：
你：讲个笑话
OpenClaw：好的，听我说...

你：猜谜语
OpenClaw：什么东西...

```
### 9.4.5 限制和注意事项

**功能限制**：
```

⚠️ QQ机器人有以下限制：

- 消息频率限制
- 功能权限限制
- 审核要求严格
- 部分API需要申请
- 目前不支持主动发送消息（2026.2.6测试）

```
**注意事项**：
```

✅ 遵守平台规则
✅ 不发送违规内容
✅ 合理使用API
✅ 及时响应用户
✅ 定期检查服务状态

常见问题：

1. **机器人不回复消息**：
   - 检查IP白名单是否正确
   - 检查机器人ID和密钥是否正确
   - 查看Gateway服务是否运行
   - 检查服务器日志

2. **配置后无法连接**：
   - 确认已添加为测试用户
   - 确认已添加机器人好友
   - 重启Gateway服务
   - 检查防火墙设置

3. **消息延迟**：
   - 检查网络连接
   - 检查服务器负载
   - 考虑升级服务器配置

9.5 Discord Bot配置（参考）

> ⚠️ **过时提示**：本节内容编写于2026年1月，当时OpenClaw还叫Clawbot/Moltbot。虽然部分命令已过时，但配置流程仍可作为参考。

9.5.1 Discord机器人介绍

Discord的优势：

1. **国际化平台**
   - 全球用户基础
   - 多语言支持
   - 社区活跃

2. **开发友好**
   - API完善
   - 文档详细
   - 权限灵活

3. **功能丰富**
   - 支持语音频道
   - 支持富文本
   - 支持自定义表情

适用场景：
- ✅ 国际团队协作
- ✅ 游戏社区
- ✅ 开源项目
- ✅ 技术交流

```
### 9.5.2 创建Discord机器人

**步骤1：访问开发者门户**
```

https://discord.com/developers/applications

步骤2：创建应用

1. 点击"New Application"
2. 输入应用名称（如：My OpenClaw Bot）
3. 点击"Create"

步骤3：创建Bot

1. 在左侧菜单选择"Bot"
2. 点击"Add Bot"
3. 点击"Reset Token" → "Copy"
4. ⚠️ **保存Token**，后续无法再查看

步骤4：配置Bot权限

1. 在Bot页面下滑
2. 开启"Message Content Intent"
3. 点击"Save Changes"

步骤5：生成邀请链接

1. 在左侧菜单选择"OAuth2" → "URL Generator"
2. 在"Scopes"中勾选：`bot`
3. 在"Bot Permissions"中勾选：
   - Send Messages
   - Read Message History
4. 复制生成的URL

步骤6：邀请Bot到服务器

1. 在浏览器中打开刚才复制的URL
2. 选择你的Discord服务器
3. 点击"授权"
4. 完成验证

9.5.3 配置 OpenClaw（旧版命令参考）

> ⚠️ **注意**：以下命令使用的是旧版本的`clawdbot`命令，新版本应使用`openclaw`。

配置步骤（需要更新为新命令）：

```bash
# 旧版命令（仅供参考）
clawdbot onboard

# 新版命令（推荐）
openclaw onboard

# 配置流程：
# 1. 选择 Yes 接受风险
# 2. 选择 QuickStart
```

```
# 3. 配置模型（如 GLM 4.7）
# 4. 选择通道：Discord
# 5. 输入 Bot Token
# 6. 配置 Skills 和 Hooks
```text
**启动服务**：

```bash
# 旧版命令
clawdbot gateway --port 18789 --verbose

# 新版命令
openclaw gateway --port 18789 --verbose

# 后台运行
nohup openclaw gateway --port 18789 --verbose > /dev/null 2>&1 &
```text
**配对连接**：

```bash
# 1. 在Discord中私聊Bot，获取配对码
# 2. 停止Gateway服务（Ctrl+C）
# 3. 运行配对命令（旧版）
clawdbot pairing approve discord <Pairing code>

# 新版命令（需要确认）
openclaw pairing approve discord <Pairing code>

# 4. 重新启动Gateway
openclaw gateway --port 18789 --verbose
```text
### 9.5.4 使用Discord Bot

**私聊模式**：
```

1. 在Discord中找到你的Bot
2. 点击Bot头像
3. 点击"发送消息"
4. 直接发送消息即可

```
**群聊模式**：
```

1. 在频道中@Bot
2. 输入你的问题
3. Bot会回复你

示例：
@MyBot 今天天气怎么样？
```

```
### 9.5.5 注意事项

**命令更新**：
- 本节使用的`clawdbot`命令已过时
- 新版本统一使用`openclaw`命令
- 配置流程基本相同，但命令需要更新

**配置参考**：
- Discord的配置流程仍然有效
- Bot创建步骤没有变化
- 主要是OpenClaw 命令需要更新

**推荐做法**：
- 优先使用国内平台（飞书、QQ、企微）
- Discord适合国际团队
- 如需使用Discord，请参考最新官方文档


---

## 9.5 平台对比与选择

### 9.5.1 功能对比

| 功能 | 飞书 | 企业微信 | 钉钉 | QQ |
|------|------|---------|------|-----|
| 企业办公 | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ | ⭐⭐ |
| 即时通讯 | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ |
| 文档协作 | ⭐⭐⭐⭐⭐ | ⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐ |
| 开发友好 | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐⭐⭐ |
| 用户基础 | ⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ |
| 免费额度 | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ |

### 9.5.2 使用场景推荐

**飞书**：
```

✅ 适合场景：

- 现代化办公
- 文档协作
- 知识管理
- 团队协作
- 技术团队

❌ 不适合：

- 传统企业
- 简单需求

```
**企业微信**：
```

✅ 适合场景：

- 企业内部使用
- 需要与微信互通
- 客户服务
- 营销推广

❌ 不适合：

- 纯个人使用
- 需要复杂文档协作

```
**钉钉**：
```

✅ 适合场景：

- 企业办公
- 考勤管理
- 审批流程
- 项目管理

❌ 不适合：

- 个人娱乐
- 社交互动

```
**QQ**：
```

✅ 适合场景：

- 个人使用
- 社交互动
- 娱乐功能
- 学生群体

❌ 不适合：

- 企业办公
- 正式场合

```
### 9.5.3 多平台组合策略

**策略1：工作+生活分离**
```

工作：飞书/企业微信/钉钉
生活：QQ

优势：

- 工作生活分离
- 专注度更高
- 管理更方便

```
**策略2：全平台覆盖**
```

同时接入所有平台

优势：

- 覆盖所有用户
- 随时随地使用
- 功能互补

劣势：

- 维护成本高
- 消息分散

```
**策略3：主次搭配（推荐）**
```

主平台：飞书（日常使用）
辅平台：企业微信（客户沟通）

优势：

- 重点突出
- 成本可控
- 易于管理

```
---

## 📝 本章小结

本章学习了OpenClaw的多平台集成功能：

### 核心内容

1. **飞书Bot配置**
   - 创建飞书应用
   - 配置 OpenClaw
   - 实战案例（个人助手、项目管理）
   - 与飞书生态集成
   - 高级功能（流式输出、多Agent路由）

2. **企业微信Bot配置**
   - 注册和创建应用
   - 配置 OpenClaw
   - 实战案例（个人助手、团队协作、客户服务）
   - 手机端使用技巧

3. **钉钉Bot配置**
   - 创建钉钉应用
   - 配置 OpenClaw
   - 实战案例（工作助手、审批流程）

4. **QQ Bot配置**
   - 创建QQ机器人
   - 配置 OpenClaw
   - 实战案例（个人助手、群管理、娱乐互动）
   - 限制和注意事项

### 平台选择

- **飞书**：现代化办公、文档协作、技术团队（推荐优先）
- **企业微信**：企业办公、客户服务
- **钉钉**：考勤管理、审批流程
- **QQ**：个人使用、社交互动

### 实战技巧

- ✅ 选择合适的平台
- ✅ 合理配置权限
- ✅ 优化使用体验
- ✅ 多平台组合使用
- ✅ 遵守平台规则

### 下一步

- 学习第10章：API服务封装
- 掌握Banana绘图、Notion同步等
```

- 构建多功能AI工具箱

🎯 实战练习

练习1：配置飞书Bot
1. 注册飞书开放平台
2. 创建应用
3. 配置 OpenClaw
4. 测试文档集成

练习2：配置企业微信Bot
1. 注册企业微信
2. 创建应用
3. 配置 OpenClaw
4. 测试基本功能

练习3：多平台对比
1. 分别体验4个平台
2. 对比功能差异
3. 选择适合自己的平台

💡 常见问题

Q1：哪个平台最好用？
A：看使用场景。技术团队推荐飞书（开发友好、功能强大），企业用飞书/钉钉，个人用QQ，客户服务用企业微信。

Q2：可以同时接入多个平台吗？
A：可以，OpenClaw支持同时接入多个平台。

Q3：配置复杂吗？
A：云端部署很简单，参考官方教程即可。飞书配置最简单，支持WebSocket长连接。

Q4：免费吗？
A：平台基础功能都免费，OpenClaw也免费。

Q5：手机上能用吗？
A：可以，所有平台都支持手机端。飞书的移动端体验最好。

📚 参考资源

官方教程

飞书：
- 快速接入指南：https://cloud.tencent.com/developer/article/2626151
- 视频教程：https://cloud.tencent.com/developer/video/85055

企业微信：
- 快速接入指南：https://cloud.tencent.com/developer/article/2625147
- 视频教程：https://cloud.tencent.com/developer/video/85003

钉钉：
- 快速接入指南：https://cloud.tencent.com/developer/article/2626553
- 视频教程：https://cloud.tencent.com/developer/video/85055

QQ：
- 快速接入指南：https://cloud.tencent.com/developer/article/2626045
- 视频教程：https://cloud.tencent.com/developer/video/85003

社区资源

- OpenClaw社区：https://docs.openclaw.ai
- 交流群：扫码加入
- 问题反馈：GitHub Issues

下一章预告：第10章将学习API服务封装，包括Banana绘图集成、Notion数据同步、视频生成服务、语音合成接入等内容。

9.1.16 多机器人多 Agent 模式：打造你的 AI 助手团队

> 💡 **完整教程**：本节详细介绍如何使用多 Gateway + 多飞书机器人架构，打造专业的 AI 助手团队。

9.1.16.1 为什么需要多 Agent？

作为超级个体创业者，你可能需要不同类型的 AI 助手来处理不同的工作：

- **主助理**：使用最强大的模型（Claude Opus）处理复杂任务
- **内容创作助手**：专注于文章写作、文案创作
- **技术开发助手**：处理代码开发、技术问题
- **AI 资讯助手**：快速获取和整理 AI 行业动态

传统的单 Agent 模式需要频繁切换模型和上下文，效率低下。多 Agent 模式让你可以同时拥有多个专业助手，各司其职。

![多Agent架构](https://i-blog.csdnimg.cn/img_convert/d9d0d47052a8dbef500c9ceab133ee7e.png)

9.1.16.2 实现方案对比

方案一：单 Gateway + Bindings（不推荐）

```json
{
  "bindings": [
    {
      "agentId": "main-agent",
      "match": {
        "channel": "feishu",
        "peer": {
```

```
          "kind": "group",
          "id": "oc_xxx"
        }
      }
    }
  ]
}
```text
**问题**：
- ❌ OpenClaw 2026.2.9 的 bindings 功能不稳定
- ❌ peer.id 匹配经常失败
- ❌ 所有群组都路由到同一个 agent
- ❌ 需要 `/reset` + `/agent` 命令手动切换

#### 方案二：多 Gateway + 多飞书机器人（推荐）✅

**核心思路**：
- 创建 4 个飞书机器人应用
- 启动 4 个独立的 OpenClaw Gateway
- 每个 Gateway 连接一个飞书机器人
- 每个 Gateway 使用不同的 Agent 和模型

**优势**：
- ✅ 完全独立，互不干扰
- ✅ 直接私聊不同机器人即可切换 agent
- ✅ 不需要群组配置
- ✅ 不需要手动切换命令
- ✅ 配置清晰，易于管理
- ✅ 可以独立重启某个 Gateway

### 9.1.16.3 架构设计

#### 整体架构
```

```
┌──────────────────────────────────────────────────────────────┐
│ 飞书 (Feishu) │
├──────────────────────────────────────────────────────────────┤
│ 机器人1: 主助理 机器人2: 内容创作助手 │
│ 机器人3: 技术开发助手 机器人4: AI资讯助手 │
└──────────────────────────────────────────────────────────────┘
                              ↓ WebSocket
┌──────────────────────────────────────────────────────────────┐
│ OpenClaw Gateway 层 │
├──────────────────┬──────────────────┬──────────────────┬──────────────────┤
│ Gateway 1 │ Gateway 2 │ Gateway 3 │ Gateway 4 │
│ 端口: 18789 │ 端口: 18790 │ 端口: 18791 │ 端口: 18792 │
│ Profile: │ Profile: │ Profile: │ Profile: │
│ main- │ content- │ tech-dev │ ai-news │
│ assistant │ creator │ │ │
└──────────────────┴──────────────────┴──────────────────┴──────────────────┘
```
```

↓

Agent 层			
main-agent Claude Opus 4.6 Thinking	content-agent Claude Sonnet 4.5	tech-agent Claude Sonnet 4.5 Thinking	ainews-agent Gemini 2.5 Flash

Profile 隔离机制

使用 `--profile <name>` 参数，OpenClaw 会：
- 配置文件：`~/.openclaw-<name>/openclaw.json`
- 状态数据：`~/.openclaw-<name>/`
- 独立端口：18789, 18790, 18791, 18792
- 独立会话：完全隔离的上下文

9.1.16.4 配置步骤

第一步：创建飞书机器人应用

在飞书开放平台创建 4 个机器人应用：

1. **主助理**
   - 应用名称：主助理
   - 描述：处理复杂任务的主力助手
   - 获取 App ID 和 App Secret

2. **内容创作助手**
   - 应用名称：内容创作助手
   - 描述：专注内容创作和文案写作
   - 获取 App ID 和 App Secret

3. **技术开发助手**
   - 应用名称：技术开发助手
   - 描述：处理代码开发和技术问题
   - 获取 App ID 和 App Secret

4. **AI资讯助手**
   - 应用名称：AI资讯助手
   - 描述：快速获取 AI 行业资讯
   - 获取 App ID 和 App Secret

重要配置：
- 启用机器人能力
- 配置事件订阅：选择"长连接"模式
- 添加权限：消息接收、消息发送

第二步：配置 Agent

创建 4 个 Agent 配置目录：

```bash
mkdir -p agent-configs/{main-agent,content-agent,tech-agent,ainews-agent}
```text
为每个 Agent 创建配置文件：

**agent-configs/main-agent/USER.md**：

```markdown
```

用户信息

- 姓名：Maynor
- 职业：超级个体创业者
- 工作领域：AI 技术、内容创作、技术开发
```text
**agent-configs/main-agent/SOUL.md**：

```markdown
# Agent 身份

你是 Maynor 的主助理，负责处理各类复杂任务。使用 Claude Opus 4.6 Thinking 模型，提供最高质量的服务。
```text
类似地为其他 3 个 Agent 创建配置文件。

#### 第三步：运行配置脚本

使用自动化脚本创建多 Gateway 配置：

```bash
# 下载配置脚本
curl -O https://example.com/setup-multi-gateway.sh
chmod +x setup-multi-gateway.sh

# 运行配置脚本
./setup-multi-gateway.sh
```text
脚本会自动：
1. 停止当前 Gateway
2. 备份现有配置
3. 创建 4 个独立的 Profile 配置
4. 生成管理脚本

#### 第四步：启动所有 Gateway

```bash
# 启动所有 Gateway
./start-all-gateways.sh

# 检查状态
./check-gateways.sh

# 验证配置
./verify-setup.sh
```text
### 9.1.16.5 使用方法

#### 直接私聊机器人

这是最简单的使用方式：

1. **处理复杂任务**
```

```
   - 在飞书中搜索"主助理"机器人
   - 直接发送消息
   - 自动使用 Claude Opus 4.6 Thinking

2. **创作内容**
   - 搜索"内容创作助手"机器人
   - 发送写作需求
   - 自动使用 Claude Sonnet 4.5

3. **开发代码**
   - 搜索"技术开发助手"机器人
   - 发送技术问题
   - 自动使用 Claude Sonnet 4.5 Thinking

4. **获取资讯**
   - 搜索"AI资讯助手"机器人
   - 请求最新动态
   - 自动使用 Gemini 2.5 Flash（快速响应）

#### 在群组中使用（可选）

如果需要在群组中使用：

1. 将对应的机器人添加到群组
2. @ 机器人发送消息
3. 每个群组可以添加多个机器人，灵活切换

**建议**：
- 工作群：添加主助理 + 技术开发助手
- 内容创作群：添加内容创作助手
- 资讯群：添加 AI资讯助手

### 9.1.16.6 管理和维护

#### 日常管理

```bash
# 查看所有 Gateway 状态
./check-gateways.sh

# 查看实时日志
tail -f logs-main-assistant.log
tail -f logs-content-creator.log
tail -f logs-tech-dev.log
tail -f logs-ai-news.log

# 查看所有日志
tail -f logs-*.log
```text
#### 重启 Gateway

```bash
```
```

```
# 重启所有
./stop-all-gateways.sh
sleep 2
./start-all-gateways.sh

# 重启单个
ps aux | grep "openclaw.*--profile main-assistant"
kill <PID>
./start-main-assistant.sh
```text
#### 修改配置

```bash
# 编辑配置
vim ~/.openclaw-main-assistant/openclaw.json

# 验证配置
jq . ~/.openclaw-main-assistant/openclaw.json

# 重启生效
# (停止并重启对应的 Gateway)
```text
#### 监控资源

```bash
# 查看内存占用
ps aux | grep openclaw-gateway | awk '{print $4, $11}'

# 查看 CPU 占用
ps aux | grep openclaw-gateway | awk '{print $3, $11}'

# 查看端口占用
lsof -i :18789
lsof -i :18790
lsof -i :18791
lsof -i :18792
```text
### 9.1.16.7 实战案例

#### 案例一：内容创作工作流

**场景**：写一篇技术文章

1. **构思阶段**
   - 私聊"主助理"：讨论文章主题和大纲
   - 使用 Claude Opus 进行深度思考

2. **写作阶段**
   - 私聊"内容创作助手"：撰写文章内容
   - 使用 Claude Sonnet 快速生成

3. **代码示例**
```
```

```
   - 私聊"技术开发助手"：编写代码示例
   - 使用 Claude Sonnet Thinking 确保代码质量

4. **资讯补充**
   - 私聊"AI资讯助手"：获取最新技术动态
   - 使用 Gemini Flash 快速检索

#### 案例二：技术开发工作流

**场景**：开发一个新功能

1. **需求分析**
   - 主助理：分析需求，设计架构

2. **代码实现**
   - 技术开发助手：编写代码，调试问题

3. **文档编写**
   - 内容创作助手：编写技术文档

4. **技术调研**
   - AI资讯助手：查找相关技术资料

#### 案例三：日常工作场景

**上午 9:00 - 规划工作**
- 主助理：制定今天的工作计划

**上午 10:00 - 写作**
- 内容创作助手：撰写文章

**下午 2:00 - 开发**
- 技术开发助手：编写代码

**下午 4:00 - 学习**
- AI资讯助手：了解行业动态

**晚上 8:00 - 总结**
- 主助理：总结今天的工作

### 9.1.16.8 性能和成本

#### 资源占用

- **内存**：每个 Gateway 约 400MB
- **总内存**：4 个 Gateway 约 1.6GB
- **CPU**：空闲时几乎为 0，处理时根据任务而定
- **磁盘**：配置文件和日志约 100MB

#### 成本分析

假设使用自建 API 代理：
```

Agent	模型	用途	月使用量	月成本
main-agent	Claude Opus 4.6	复杂任务	100万 tokens	$15
content-agent	Claude Sonnet 4.5	内容创作	200万 tokens	$6
tech-agent	Claude Sonnet 4.5	技术开发	150万 tokens	$4.5
ainews-agent	Gemini 2.5 Flash	资讯获取	300万 tokens	$0
总计	-	-	750万 tokens	**$25.5**

成本优化建议：
- 简单任务使用 Gemini Flash（免费）
- 复杂任务才使用 Claude Opus
- 内容创作使用 Claude Sonnet（性价比高）

9.1.16.9 故障排查

Gateway 启动失败

症状：运行 `./start-all-gateways.sh` 后，`./check-gateways.sh` 显示进程未运行

排查步骤：

```bash
# 1. 查看日志
tail -50 logs-main-assistant.log

# 2. 检查配置
jq . ~/.openclaw-main-assistant/openclaw.json

# 3. 检查端口占用
lsof -i :18789

# 4. 运行 doctor
openclaw --profile main-assistant doctor
```text
**常见问题**：
- 配置文件格式错误：运行 `jq` 验证
- 端口被占用：更换端口或停止占用进程
- 飞书配置错误：检查 App ID 和 App Secret

#### 机器人无响应

**症状**：在飞书中 @ 机器人，没有回复

**排查步骤**：

```bash
# 1. 检查 Gateway 是否运行
./check-gateways.sh

# 2. 查看实时日志
tail -f logs-main-assistant.log
```

```
# 3. 检查飞书连接
grep "WebSocket client started" logs-main-assistant.log
```text
**常见原因**：
- Gateway 未启动：运行 `./start-all-gateways.sh`
- 飞书连接断开：检查网络，重启 Gateway
- 配置错误：验证飞书 App ID 和 Secret

#### 使用了错误的 Agent

**症状**：私聊"内容创作助手"，但使用的是 Claude Opus 模型

**原因**：配置文件中 Agent 设置错误

**解决**：

```bash
# 检查配置
jq '.agents.list[0].id, .agents.list[0].model.primary' \
  ~/.openclaw-content-creator/openclaw.json

# 应该输出：
# "content-agent"
# "local-antigravity/claude-sonnet-4-5"
```text
### 9.1.16.10 高级技巧

#### 技巧一：使用 tmux 管理

```bash
# 创建 tmux 会话
tmux new -s openclaw

# 分割窗口
Ctrl+b %  # 垂直分割
Ctrl+b "  # 水平分割

# 在不同窗口中运行不同的 Gateway
./start-main-assistant.sh
./start-content-creator.sh
./start-tech-dev.sh
./start-ai-news.sh

# 查看所有日志
tail -f logs-*.log
```text
#### 技巧二：配置开机自启动

使用 launchd（macOS）：

```bash
```

```
# 创建 plist 文件
cat > ~/Library/LaunchAgents/com.openclaw.main-assistant.plist << 'EOF'
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.openclaw.main-assistant</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/local/bin/openclaw</string>
    <string>--profile</string>
    <string>main-assistant</string>
    <string>gateway</string>
    <string>run</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>KeepAlive</key>
  <true/>
</dict>
</plist>
EOF

# 加载服务
launchctl load ~/Library/LaunchAgents/com.openclaw.main-assistant.plist
```text
#### 技巧三：日志轮转

```bash
# 创建日志轮转脚本
cat > rotate-logs.sh << 'EOF'
#!/bin/zsh
for log in logs-*.log; do
  if [ -f "$log" ] && [ $(stat -f%z "$log") -gt 10485760 ]; then
    mv "$log" "$log.$(date +%Y%m%d_%H%M%S)"
    touch "$log"
  fi
done
EOF

chmod +x rotate-logs.sh

# 添加到 crontab（每小时执行）
crontab -e
# 添加：0 * * * * /path/to/rotate-logs.sh
```text
### 9.1.16.11 总结

多 Gateway + 多飞书机器人的方案是目前最稳定、最简单的多 Agent 实现方式：
```
```

核心优势：
- ✅ 直接私聊不同机器人，自动使用对应 agent
- ✅ 完全独立，互不干扰
- ✅ 不需要复杂的 bindings 配置
- ✅ 不需要手动切换命令
- ✅ 配置清晰，易于管理

适用场景：
- 超级个体创业者
- 需要多个专业助手
- 不同任务使用不同模型
- 追求稳定性和可靠性

下一步：
1. 创建飞书机器人应用
2. 运行配置脚本
3. 启动所有 Gateway
4. 开始使用你的 AI 助手团队！

9.1.17 多 Agent 配置（传统方式）

> ⚠️ **注意**：本节介绍的是传统的单 Gateway + Bindings 方式，推荐使用上面的多 Gateway 方案。

什么是多 Agent？

多 Agent 配置允许：
- 每个飞书机器人使用不同的 Agent
- 每个 Agent 使用不同的模型
- 每个 Agent 使用独立的工作空间
- 每个 Agent 有独立的配置和上下文

9.1.17.1 配置结构（传统方式）

```json
{
  "agents": {
    "list": [
      {
        "id": "agent-id",
        "workspace": "/path/to/workspace",
        "model": {
          "primary": "provider/model"
        }
      }
    ],
    "defaults": {
      "compaction": { "mode": "safeguard" },
      "maxConcurrent": 4
    }
  },
```

```
  "channels": {
    "feishu": {
      "accounts": {
        "bot-name": { ... }
      }
    }
  },
  "bindings": [
    {
      "agentId": "agent-id",
      "match": {
        "channel": "feishu",
        "peer": {
          "kind": "dm",
          "id": "ou_user_id"
        }
      }
    }
  ]
}
```text
### 9.1.17.2 实战案例：4个专业助手（传统方式）

**场景**：一人公司，需要不同的专业助手处理不同任务。

**配置示例**：

```json
{
  "agents": {
    "list": [
      {
        "id": "main-agent",
        "workspace": "/Users/username/clawd",
        "model": {
          "primary": "anthropic/claude-sonnet-4"
        }
      },
      {
        "id": "content-agent",
        "workspace": "/Users/username/clawd/content",
        "model": {
          "primary": "anthropic/claude-sonnet-4"
        }
      },
      {
        "id": "tech-agent",
        "workspace": "/Users/username/clawd/tech",
        "model": {
          "primary": "anthropic/claude-sonnet-4"
        }
      },
```

```
      {
        "id": "ainews-agent",
        "workspace": "/Users/username/clawd/ainews",
        "model": {
          "primary": "google/gemini-2-flash"
        }
      }
    ],
    "defaults": {
      "compaction": { "mode": "safeguard" },
      "maxConcurrent": 4,
      "subagents": { "maxConcurrent": 8 }
    }
  },
  "channels": {
    "feishu": {
      "accounts": {
        "main-assistant": {
          "appId": "cli_main_xxx",
          "appSecret": "xxx",
          "botName": "主助理"
        },
        "content-creator": {
          "appId": "cli_content_xxx",
          "appSecret": "xxx",
          "botName": "内容创作助手"
        },
        "tech-dev": {
          "appId": "cli_tech_xxx",
          "appSecret": "xxx",
          "botName": "技术开发助手"
        },
        "ai-news": {
          "appId": "cli_news_xxx",
          "appSecret": "xxx",
          "botName": "AI资讯助手"
        }
      }
    }
  },
  "bindings": [
    {
      "agentId": "main-agent",
      "match": {
        "channel": "feishu",
        "peer": { "kind": "dm", "id": "ou_xxx1" }
      }
    },
    {
      "agentId": "content-agent",
      "match": {
        "channel": "feishu",
```

```
        "peer": { "kind": "dm", "id": "ou_xxx2" }
      }
    },
    {
      "agentId": "tech-agent",
      "match": {
        "channel": "feishu",
        "peer": { "kind": "dm", "id": "ou_xxx3" }
      }
    },
    {
      "agentId": "ainews-agent",
      "match": {
        "channel": "feishu",
        "peer": { "kind": "dm", "id": "ou_xxx4" }
      }
    }
  ]
}
```text
### 9.1.17.3 获取用户 ID（传统方式）

**方法1：通过日志获取（推荐）**

```bash
# 1. 启动网关并查看日志
openclaw gateway restart
openclaw logs --follow

# 2. 在飞书中给每个机器人发送消息

# 3. 在日志中查找 open_id
# 格式：ou_xxxxxxxxxxxxxxxx
```text
**日志示例**：
```

[feishu] Received message from ou_18d36d8a49c010dfe20ace2a29250c04
[feishu] Bot: 主助理
```

方法2：通过配对请求获取

```bash
openclaw pairing list feishu

# 输出示例：
# Pending pairing requests:
# - Code: ABC123, User: ou_xxx, Bot: 主助理
```text
### 9.1.17.4 配置步骤（传统方式）

**步骤1：创建工作空间目录**

```bash
mkdir -p /Users/username/clawd/content
mkdir -p /Users/username/clawd/tech
mkdir -p /Users/username/clawd/ainews
```text
**步骤2：获取所有用户 ID**

按照上面的方法，获取每个机器人对应的用户 ID。

**步骤3：更新配置文件**

将获取到的用户 ID 填入 `bindings` 部分。

**步骤4：应用配置**

```bash
# 备份现有配置
cp ~/.openclaw/openclaw.json ~/.openclaw/openclaw.json.backup

# 应用新配置
cp your-config.json ~/.openclaw/openclaw.json

# 验证配置
openclaw doctor

# 重启网关
openclaw gateway restart
```text
**步骤5：验证运行**

```bash
# 查看 Agent 状态
openclaw doctor

# 应该看到：
# Agents: main-agent (default), content-agent, tech-agent, ainews-agent
# Session store: 4 entries
```

```
# 查看日志
openclaw logs --follow | grep bindings

# 应该看到：
# [bindings] Matched agent: main-agent for user ou_xxx
```text
### 9.1.17.5 配置注意事项

**⚠️ 重要：agents.list 配置限制**

这是最常见的配置错误！`agents.list` 中的每个 Agent 只能包含以下字段：

```json
{
  "id": "agent-id",           // ✅ Agent 标识符（必填）
  "workspace": "/path",       // ✅ 工作空间路径（必填）
  "model": { "primary": "" }  // ✅ 使用的模型（可选）
  // ❌ 不能包含 compaction
  // ❌ 不能包含 maxConcurrent
  // ❌ 不能包含 subagents
  // ❌ 不能包含 models
}
```text
**错误示例（会导致配置验证失败）**：
```json
{
  "agents": {
    "list": [
      {
        "id": "main-agent",
        "workspace": "/path",
        "compaction": { "mode": "safeguard" },  // ❌ 错误！
        "maxConcurrent": 4                      // ❌ 错误！
      }
    ]
  }
}
```text
**正确示例**：
```json
{
  "agents": {
    "list": [
      {
        "id": "main-agent",
        "workspace": "/path",
        "model": { "primary": "provider/model" }  // ✅ 正确
      }
    ],
    "defaults": {
      "compaction": { "mode": "safeguard" },  // ✅ 在这里配置
      "maxConcurrent": 4,                     // ✅ 在这里配置
```

```
      "subagents": { "maxConcurrent": 8 }     // ✅ 在这里配置
    }
  }
}
```text
**2. 通用配置必须放在 agents.defaults**：

所有 Agent 共享的配置项必须放在 `agents.defaults` 中，包括：
- `compaction` - 上下文压缩策略
- `maxConcurrent` - 最大并发数
- `subagents` - 子 Agent 配置
- `models` - 额外的模型配置

```json
{
  "defaults": {
    "model": {
      "primary": "default-provider/default-model"
    },
    "workspace": "/default/workspace",
    "compaction": { "mode": "safeguard" },
    "maxConcurrent": 4,
    "subagents": { "maxConcurrent": 8 }
  }
}
```text
**3. Bindings 顺序很重要**：

OpenClaw 会按顺序匹配 bindings，第一个匹配的规则会被使用。

```json
"bindings": [
  // 1. 最具体的匹配（特定用户）
  { "agentId": "main-agent", "match": { "peer": { "id": "ou_xxx" } } },
  // 2. 较具体的匹配（特定群组）
  { "agentId": "tech-agent", "match": { "peer": { "kind": "group" } } },
  // 3. 最后是默认匹配
  { "agentId": "default-agent", "match": { "channel": "feishu" } }
]
```text
**4. 用户 ID 是唯一的**：

每个飞书用户只能绑定到一个 Agent。

### 故障排查

**问题1：配置验证失败 - agents.list 包含不支持的字段**

```bash
# 错误信息
Config invalid
File: ~/.openclaw/openclaw.json
```

```text
Problem:
- agents.list.0: Unrecognized keys: "compaction", "maxConcurrent"
- agents.list.1: Unrecognized keys: "compaction", "maxConcurrent"
- agents.list.2: Unrecognized keys: "compaction", "maxConcurrent"
- agents.list.3: Unrecognized keys: "compaction", "maxConcurrent"
Run: openclaw doctor --fix
```text
**原因**：`agents.list` 中的 Agent 配置包含了只能在 `agents.defaults` 中使用的字段。

**解决方案**：

```bash
# 方法1：自动修复（推荐）
openclaw doctor --fix

# 方法2：手动修复
# 编辑配置文件，将 compaction 和 maxConcurrent 从 agents.list 移到 agents.defaults
```text
**修复前**：
```json
{
  "agents": {
    "list": [
      {
        "id": "main-agent",
        "workspace": "/path",
        "compaction": { "mode": "safeguard" },  // ❌ 错误位置
        "maxConcurrent": 4                      // ❌ 错误位置
      }
    ]
  }
}
```text
**修复后**：
```json
{
  "agents": {
    "list": [
      {
        "id": "main-agent",
        "workspace": "/path",
        "model": { "primary": "provider/model" }
      }
    ],
    "defaults": {
      "compaction": { "mode": "safeguard" },  // ✅ 正确位置
      "maxConcurrent": 4                      // ✅ 正确位置
    }
  }
}
```text
**验证修复**：
```
```

```bash
# 验证配置
openclaw doctor

# 应该看到：
# ✅ Config valid
# ✅ 4 agents configured
# ✅ 4 bindings configured
```text
**问题2：Bindings 不生效**

```bash
# 检查用户 ID 是否正确
openclaw logs --follow | grep "ou_"

# 查看 bindings 匹配情况
openclaw logs --follow | grep bindings
```text
**问题3：找不到用户 ID**

```bash
# 使用 debug 级别日志
openclaw logs --follow --level debug

# 或查看配对请求
openclaw pairing list feishu
```text
**问题4：配置修改后运行 openclaw doctor 报错**

```bash
# 错误信息
Unknown config keys:
- agents.list[0].compaction
- agents.list[0].maxConcurrent
- agents.list[1].compaction
- agents.list[1].maxConcurrent
...

Run "openclaw doctor --fix" to remove these keys.
```text
**解决方案**：
```bash
# 运行自动修复
openclaw doctor --fix

# 验证配置
openclaw doctor

# 重启网关
openclaw gateway restart

# 查看状态
```

```
openclaw gateway status
```text
**问题5：版本不匹配警告**

```bash
# 警告信息
Config was last written by a newer OpenClaw (2026.2.6-3);
current version is 2026.2.1-zh.3.
Run "openclaw doctor --fix" to apply changes.
```text
**说明**：这是正常的版本提示，不影响使用。如果想消除警告：
```bash
openclaw doctor --fix
```text
### 9.1.17.6 配置对比

| 特性 | 单 Agent 模式 | 多 Agent 模式 |
|------|--------------|--------------|
| 配置复杂度 | 简单 | 复杂 |
| 模型选择 | 所有机器人相同 | 每个机器人不同 |
| 工作空间 | 共享 | 隔离 |
| 需要 bindings | ❌ | ✅ |
| 需要用户 ID | ❌ | ✅ |
| 适用场景 | 简单使用 | 专业分工 |

### 9.1.17.7 使用建议

**推荐使用多 Agent 的场景**：
- ✅ 需要不同机器人使用不同模型
- ✅ 需要隔离工作空间
- ✅ 需要独立配置和上下文
- ✅ 专业分工明确

**推荐使用单 Agent 的场景**：
- ✅ 配置简单易维护
- ✅ 所有机器人使用相同模型
- ✅ 不需要隔离工作空间
- ✅ 快速开始使用

---

## 9.1.18 本地多 Agent 管理（无需绑定 IM 平台）

> 💡 **重要提示**：多 Agent 管理不仅可以用于飞书等 IM 平台，也完全支持本地使用。如果你不需要绑定飞书机器人，可以通过 Web UI、命令行或 TUI 界面直接使用多个 Agent。

![本地多Agent管理界面 - Web UI/命令行/TUI三种方式]
(https://upload.maynor1024.live/file/1770944487857_image-20260213090121654.png)

### 本地使用方式

OpenClaw 提供了多种本地使用方式，无需配置任何 IM 平台：
```
```

方式一：Web UI（推荐）

```bash
# 打开 Web 界面
openclaw dashboard

# 或直接访问
http://127.0.0.1:18789/?token=你的token
```text
**优势**：
- ✅ 图形化界面，操作直观
- ✅ 支持文件上传和下载
- ✅ 实时显示 Token 消耗
- ✅ 支持多轮对话历史

#### 方式二：命令行对话

```bash
# 直接发送消息
openclaw agent --message "你好，帮我分析一下这个项目"

# 使用管道输入
echo "帮我总结这个文件的内容" | openclaw agent --message

# 指定输出文件
openclaw agent --message "生成项目文档" --output docs.md
```text
**优势**：
- ✅ 快速执行单次任务
- ✅ 适合脚本自动化
- ✅ 可以集成到工作流中

#### 方式三：TUI 终端界面

```bash
# 启动终端交互界面
openclaw tui
```text
**优势**：
- ✅ 终端内交互式对话
- ✅ 支持多轮对话
- ✅ 适合服务器环境使用

### 本地多 Agent 配置

配置文件位置：`~/.openclaw/openclaw.json`

**配置示例**：

```json
{
```

```
  "agents": {
    "list": [
      {
        "id": "main-agent",
        "workspace": "/Users/username/work",
        "model": { "primary": "anthropic/claude-sonnet-4" }
      },
      {
        "id": "content-agent",
        "workspace": "/Users/username/content",
        "model": { "primary": "anthropic/claude-sonnet-4" }
      },
      {
        "id": "code-agent",
        "workspace": "/Users/username/code",
        "model": { "primary": "deepseek/deepseek-chat" }
      },
      {
        "id": "research-agent",
        "workspace": "/Users/username/research",
        "model": { "primary": "google/gemini-2-flash" }
      }
    ],
    "defaults": {
      "compaction": { "mode": "safeguard" },
      "maxConcurrent": 4,
      "subagents": { "maxConcurrent": 8 }
    }
  }
}
```text
**配置说明**：

1. **agents.list**：定义所有可用的 Agent
   - `id`：Agent 标识符（必填）
   - `workspace`：工作空间路径（必填）
   - `model.primary`：使用的模型（可选）

2. **agents.defaults**：所有 Agent 共享的配置
   - `compaction`：上下文压缩策略
   - `maxConcurrent`：最大并发数
   - `subagents`：子 Agent 配置

### Agent 管理命令

#### 列出所有 Agent

```bash
openclaw agents list

# 输出示例：
# Available agents:
```

```
# - main-agent (default)
#   Workspace: /Users/username/work
#   Model: anthropic/claude-sonnet-4
# - content-agent
#   Workspace: /Users/username/content
#   Model: anthropic/claude-sonnet-4
# - code-agent
#   Workspace: /Users/username/code
#   Model: deepseek/deepseek-chat
# - research-agent
#   Workspace: /Users/username/research
#   Model: google/gemini-2-flash
```text
#### 切换 Agent

```bash
# 切换到指定 Agent
openclaw agents switch content-agent

# 输出：
# Switched to agent: content-agent
# Workspace: /Users/username/content
# Model: anthropic/claude-sonnet-4
```text
#### 查看当前 Agent

```bash
# 查看当前使用的 Agent
openclaw agents current

# 输出：
# Current agent: content-agent
# Workspace: /Users/username/content
# Model: anthropic/claude-sonnet-4
```text
#### 查看 Agent 配置

```bash
# 查看指定 Agent 的配置
openclaw agents config content-agent

# 查看当前 Agent 的配置
openclaw agents config
```text
#### 查看 Agent 状态

```bash
# 查看所有 Agent 的状态
openclaw doctor

# 输出示例：
# ✅ Config valid
```

```
# ✅ 4 agents configured
# ✅ Gateway running
# ✅ Session store: 12 entries
```text
### 实战案例：4个专业助手

**场景**：个人开发者，需要不同的专业助手处理不同任务。

**配置步骤**：

**步骤1：创建工作空间目录**

```bash
mkdir -p ~/work/main
mkdir -p ~/work/content
mkdir -p ~/work/code
mkdir -p ~/work/research
```text
**步骤2：编辑配置文件**

```bash
# 备份现有配置
cp ~/.openclaw/openclaw.json ~/.openclaw/openclaw.json.backup

# 编辑配置
nano ~/.openclaw/openclaw.json
```text
将上面的配置示例粘贴进去，修改路径为你的实际路径。

**步骤3：验证配置**

```bash
# 验证配置是否正确
openclaw doctor

# 应该看到：
# ✅ Config valid
# ✅ 4 agents configured
```text
**步骤4：重启网关**

```bash
# 重启网关使配置生效
openclaw gateway restart

# 查看状态
openclaw gateway status
```text
**步骤5：使用不同的 Agent**

```bash
# 使用主助手处理通用任务
```

```
openclaw agents switch main-agent
openclaw agent --message "帮我整理今天的待办事项"

# 使用内容助手创作文章
openclaw agents switch content-agent
openclaw agent --message "帮我写一篇关于 AI 的文章"

# 使用代码助手开发项目
openclaw agents switch code-agent
openclaw agent --message "帮我优化这段 Python 代码"

# 使用研究助手搜集资料
openclaw agents switch research-agent
openclaw agent --message "帮我搜集关于量子计算的最新研究"
```text
### 使用场景对比

| 场景 | 推荐方式 | Agent 配置 | 优势 |
|------|---------|-----------|------|
| 个人本地使用 | Web UI + 多 Agent | 不同任务用不同 Agent | 工作空间隔离，模型灵活 |
| 团队协作 | 飞书 + 多 Agent | 不同机器人绑定不同 Agent | 团队成员各用各的助手 |
| 快速测试 | 命令行 + 单 Agent | 使用默认 Agent | 配置简单，快速上手 |
| 服务器环境 | TUI + 多 Agent | 不同项目用不同 Agent | 终端内交互，资源隔离 |

### 典型工作流

**场景：一人公司的日常工作流**

```bash
# 早上：使用主助手查看日程
openclaw agents switch main-agent
openclaw agent --message "显示今天的日程安排"

# 上午：使用代码助手开发项目
openclaw agents switch code-agent
openclaw agent --message "帮我实现用户登录功能"

# 中午：使用研究助手学习新技术
openclaw agents switch research-agent
openclaw agent --message "搜集 Rust 语言的学习资料"

# 下午：使用内容助手写文章
openclaw agents switch content-agent
openclaw agent --message "写一篇关于今天开发经验的博客"

# 晚上：使用主助手总结一天
openclaw agents switch main-agent
openclaw agent --message "生成今日工作总结"
```text
### 配置技巧

**技巧1：为不同任务使用不同模型**
```
```

```json
{
  "agents": {
    "list": [
      {
        "id": "chat-agent",
        "workspace": "/Users/username/chat",
        "model": { "primary": "anthropic/claude-sonnet-4" }
      },
      {
        "id": "code-agent",
        "workspace": "/Users/username/code",
        "model": { "primary": "deepseek/deepseek-chat" }
      },
      {
        "id": "fast-agent",
        "workspace": "/Users/username/fast",
        "model": { "primary": "google/gemini-2-flash" }
      }
    ]
  }
}
```text
**说明**：
- Claude Sonnet 4：通用对话和复杂任务
- DeepSeek：代码生成和技术问题
- Gemini Flash：快速响应和简单任务

**技巧2：使用别名简化切换**

```bash
# 在 ~/.zshrc 或 ~/.bashrc 中添加别名
alias oc-main='openclaw agents switch main-agent'
alias oc-code='openclaw agents switch code-agent'
alias oc-content='openclaw agents switch content-agent'
alias oc-research='openclaw agents switch research-agent'

# 使用别名快速切换
oc-code
openclaw agent --message "帮我写一个排序算法"
```text
**技巧3：为每个 Agent 配置独立的 Skills**

```bash
# 为代码助手安装开发相关的 Skills
openclaw agents switch code-agent
openclaw skill install github-integration
openclaw skill install code-review

# 为内容助手安装写作相关的 Skills
openclaw agents switch content-agent
```

```
openclaw skill install grammar-check
openclaw skill install seo-optimizer
```text
### 常见问题

**问题1：切换 Agent 后工作空间没变**

```bash
# 检查当前 Agent
openclaw agents current

# 检查配置
openclaw agents config

# 重启网关
openclaw gateway restart
```text
**问题2：找不到 Agent**

```bash
# 列出所有 Agent
openclaw agents list

# 检查配置文件
cat ~/.openclaw/openclaw.json | grep -A 5 "agents"
```text
**问题3：Agent 配置验证失败**

```bash
# 运行诊断
openclaw doctor

# 自动修复
openclaw doctor --fix
```text
### 最佳实践

1. **工作空间隔离**
   - 为每个 Agent 创建独立的工作空间
   - 避免不同任务的文件混在一起

2. **模型选择**
   - 根据任务类型选择合适的模型
   - 代码任务用 DeepSeek，通用任务用 Claude

3. **定期备份**
   - 定期备份配置文件
   - 使用版本控制管理配置

4. **命名规范**
   - Agent ID 使用有意义的名称
   - 工作空间路径清晰明确
```
```

5. **资源管理**
   - 合理设置 maxConcurrent
   - 定期清理不用的会话

9.12 OpenClaw Manager - 可视化管理工具

> 💡 **现代化管理界面**：OpenClaw Manager 是一个基于 React + Tailwind CSS 的 Web 管理界面，用于可视化管理多个 OpenClaw Gateway 实例。

9.12.1 为什么需要 OpenClaw Manager？

当你使用多 Gateway 架构（每个飞书机器人对应一个独立的 Gateway 实例）时，传统的命令行管理方式会变得繁琐。OpenClaw Manager 提供了：

核心价值：
- 📊 **实时监控**：一目了然查看所有 Gateway 的运行状态
- 🎮 **一键控制**：启动/停止/重启服务，无需记忆命令
- ➕ **图形化创建**：通过表单创建新 Gateway，无需手动编辑配置
- ✏️ **在线编辑**：可视化编辑 Gateway 配置和 Agent 人格
- ⚙️ **保活配置**：一键配置 launchd 保活服务
- 📝 **日志查看**：实时查看每个服务的运行日志
- 💻 **美观界面**：现代化设计，响应式布局

9.12.2 功能特性

1. 自动发现 Gateway 实例 🔍

系统会自动扫描 `~/.openclaw-*` 目录，读取配置文件并显示所有 Gateway 实例。

特性：
- 自动读取端口、模型、Agent 信息
- 缓存机制（1分钟 TTL）提升性能
- 支持手动刷新发现

2. 创建新 Gateway ➕

通过图形界面创建新的 Gateway 实例，无需手动编辑配置文件。

配置项：

基础信息：
- Profile ID：唯一标识符（如 `my-assistant`）
- 机器人名称：显示名称（如 `我的助手`）
- 端口号：Gateway 监听端口（建议 18789-18799）

Agent 配置：
- Agent ID：Agent 标识符（如 `main-agent`）
- AI 模型：

  - 预设模型：Claude Opus 4.6、Claude Sonnet 4.5、Gemini 2.5 Pro 等
  - 自定义模型：输入任意模型 ID（如 `gpt-4o`, `deepseek-chat`）

飞书配置：
- App ID：飞书应用 ID（`cli_xxxxxxxxxxxxxxxx`）
- App Secret：飞书应用密钥

人格设定 📝：
- SOUL.md 编辑器：使用 Markdown 定义 Agent 的角色、性格、专业领域、回答风格

示例 SOUL.md：

```markdown
# 技术顾问 Agent

## 角色定位
你是一个资深的技术顾问，专注于软件架构和系统设计。

## 性格特点
- 严谨、专业
- 注重细节和最佳实践
- 善于分析复杂问题

## 专业领域
- 微服务架构
- 云原生技术
- DevOps 实践
- 性能优化

## 回答风格
- 先理解需求，再提供方案
- 给出具体可行的建议
- 必要时提供代码示例和架构图
- 考虑可扩展性和维护性
```text
#### 3. 编辑 Gateway ✏️

修改现有 Gateway 的配置和人格设定。

**可修改项**：
- 机器人名称
- 端口号
- Agent ID
- AI 模型（预设或自定义）
- 飞书 App ID 和 Secret（可选）
- SOUL.md 人格设定

**注意事项**：
- Profile ID 不可修改
- 飞书密钥留空则不修改
- 修改后需要重启 Gateway 才能生效

#### 4. 删除 Gateway 🗑️

完全移除 Gateway 实例及其所有配置。

**删除内容**：
- Gateway 配置文件
- Agent 配置目录
- SOUL.md 人格文件
- launchd 保活配置（如果存在）

⚠️ **警告**：删除操作不可恢复，建议先备份重要配置。

#### 5. 服务控制 🎮

**批量操作**：
- ⚙️ 配置保活：配置 launchd 保活服务（开机自启、崩溃重启）
- ▶️ 启动所有：启动所有 Gateway 实例
- ⏹️ 停止所有：停止所有 Gateway 实例
- 🔄 重启所有：重启所有 Gateway 实例

**单个操作**：
- ✏️ 编辑：编辑 Gateway 配置
- 🗑️ 删除：删除 Gateway
- 📝 日志：查看运行日志

#### 6. 实时状态监控 📊

**显示信息**：
- 运行状态（运行中/已停止）
- 端口号
- 使用的 AI 模型
- launchd 保活状态

**状态指示**：
- 🟢 绿色：运行中
- 🔴 红色：已停止
- ⚪ 灰色：未知

**自动刷新**：每 10 秒自动刷新状态，可手动点击"刷新状态"按钮。

### 9.12.3 安装和使用

#### 安装步骤

```bash
# 1. 克隆项目
git clone https://github.com/xianyu110/openclaw-manager.git
cd openclaw-manager

# 2. 安装依赖
npm install
```

```
# 3. 启动服务（前端 + 后端）
npm start
```text
应用将在以下地址启动：
- 前端：http://localhost:3000
- 后端 API：http://localhost:3001

#### 首次使用

1. **启动应用**
   ```bash
   npm start
```

2. **打开浏览器**
   访问 http://localhost:3000

3. **配置保活服务**
   - 点击"⚙️ 配置保活"按钮
   - 等待配置完成
   - 服务将自动开机启动并在崩溃后重启

日常操作

查看服务状态：

- 界面会自动每 10 秒刷新状态
- 点击"刷新状态"按钮手动刷新
- 绿色指示灯表示运行中，红色表示已停止

控制服务：

- 启动所有：一键启动所有 Gateway
- 停止所有：一键停止所有 Gateway
- 重启所有：一键重启所有 Gateway
- 单个控制：在服务卡片中点击"重启"按钮

查看日志：

- 点击服务卡片中的"查看日志"按钮
- 显示最近 100 行日志
- 支持实时刷新

9.12.4 使用场景

场景 1：创建专业领域助手

需求：创建一个专注于前端开发的技术助手

步骤：

1. 点击"➕ 新建 Gateway"
2. 填写基础信息：
   - Profile ID: `frontend-expert`
   - 机器人名称: `前端专家`
   - 端口: `18793`

3. 配置 Agent：
   - Agent ID: `frontend-agent`
   - 模型: `Claude Sonnet 4.5 Thinking`
4. 展开人格编辑器，定义专业领域：

前端开发专家

角色定位
你是一个资深的前端开发工程师，精通现代前端技术栈。

专业领域
- React / Vue / Angular
- TypeScript
- Webpack / Vite
- CSS-in-JS / Tailwind CSS
- 性能优化
- 浏览器兼容性

回答风格
- 提供最新的最佳实践
- 给出可运行的代码示例
- 考虑性能和可维护性
- 推荐合适的工具和库
```text
5. 填写飞书配置
6. 点击"创建"

#### 场景 2：使用自定义模型

**需求**：使用 OpenAI 的 GPT-4o 模型

**步骤**：
1. 在创建或编辑 Gateway 时
2. 勾选"使用自定义模型"
3. 输入模型 ID: `gpt-4o`
4. 确保在 OpenClaw 主配置中已设置 OpenAI API Key

**支持的自定义模型**：
- OpenAI: `gpt-4o`, `gpt-4o-mini`, `gpt-4-turbo`
- Anthropic: `claude-3-opus-20240229`, `claude-3-sonnet-20240229`
- Google: `gemini-2.5-pro`, `gemini-2.5-flash`
- DeepSeek: `deepseek-chat`, `deepseek-coder`
- 其他兼容 OpenAI API 的模型

#### 场景 3：创建多个专业助手

**需求**：为不同团队创建专属助手

**助手配置**：

1. **产品经理助手**
   - Profile: `product-manager`
   - 模型: `Claude Opus 4.6`
   - 人格: 注重用户体验、数据分析、产品规划

2. **设计师助手**
```

```
   - Profile: `designer`
   - 模型: `Claude Sonnet 4.5`
   - 人格: 关注视觉设计、用户界面、交互体验

3. **运维工程师助手**
   - Profile: `devops-engineer`
   - 模型: `Claude Sonnet 4.5 Thinking`
   - 人格: 专注系统稳定性、自动化、监控告警

4. **数据分析师助手**
   - Profile: `data-analyst`
   - 模型: `Gemini 2.5 Pro`
   - 人格: 擅长数据处理、可视化、统计分析

### 9.12.5 高级技巧

#### 1. 人格设定最佳实践

**结构化定义**:

```markdown
# Agent 名称

## 角色定位
明确定义 Agent 的角色和定位

## 性格特点
- 列出 3-5 个核心性格特点
- 保持一致性

## 专业领域
- 列出专业技能
- 明确擅长的领域

## 回答风格
- 描述回答的方式
- 设定语气和风格

## 工作流程
1. 步骤化的工作方式
2. 确保逻辑清晰
3. 提供可操作的建议

## 限制和边界
- 明确不擅长的领域
- 设定合理的期望
```text
#### 2. 模型选择建议

**Claude Opus 4.6**:
- 最强推理能力
- 适合复杂问题分析
```

- 成本较高

Claude Sonnet 4.5：
- 平衡性能和成本
- 适合日常对话
- 推荐用于大多数场景

Claude Sonnet 4.5 Thinking：
- 增强的思考过程
- 适合需要深度分析的场景
- 会显示思考步骤

Gemini 2.5 Flash：
- 响应速度快
- 成本低
- 适合简单查询和快速响应

DeepSeek Chat：
- 国产模型，成本极低
- 中文能力强
- 适合日常对话和简单任务

3. 端口分配建议

推荐范围：18789-18799

示例分配：
- 18789: 主助理
- 18790: 内容创作
- 18791: 技术开发
- 18792: 数据分析
- 18793: 产品设计
- 18794: 运维支持
- 18795: 客户服务
- 18796-18799: 预留

9.12.6 故障排查

问题 1：创建 Gateway 失败

可能原因：
- Profile ID 已存在
- 端口已被占用
- 缺少必填字段

解决方法：
1. 检查错误提示
2. 使用不同的 Profile ID
3. 选择未被占用的端口
4. 确保所有必填字段已填写

问题 2：Gateway 无法启动

可能原因：
- 配置文件格式错误
- 飞书账号配置错误
- 端口被其他程序占用

解决方法：
1. 检查配置文件语法
2. 验证飞书 App ID 和 Secret
3. 使用 `lsof -i :端口号` 检查端口占用
4. 查看日志文件排查错误

问题 3：人格设定不生效

可能原因：
- SOUL.md 文件未保存
- Gateway 未重启
- Agent ID 不匹配

解决方法：
1. 确认 SOUL.md 已保存
2. 重启 Gateway 服务
3. 检查 Agent ID 是否正确
4. 查看 `~/.openclaw-{profile}/agent-configs/{agent}/SOUL.md`

问题 4：后端无法连接

```bash
# 检查端口占用
lsof -i :3001

# 手动启动后端
npm run server
```text
#### 问题 5：前端无法访问

```bash
# 检查端口占用
lsof -i :3000

# 清除缓存重新启动
rm -rf node_modules/.vite
npm start
```text
### 9.12.7 API 文档

#### 状态查询

**GET /api/status**

获取所有服务的状态信息

```
响应示例：
```json
{
  "services": [
    {
      "id": "main-assistant",
      "name": "主助理",
      "port": 18789,
      "status": "running",
      "model": "Claude Opus 4.6",
      "launchd": true
    }
  ]
}
```
```

批量操作

- `POST /api/start-all` - 启动所有 Gateway 服务
- `POST /api/stop-all` - 停止所有 Gateway 服务
- `POST /api/restart-all` - 重启所有 Gateway 服务
- `POST /api/setup-launchd` - 配置 launchd 保活服务

单个服务操作

- `POST /api/start/:serviceId` - 启动指定的 Gateway 服务
- `POST /api/stop/:serviceId` - 停止指定的 Gateway 服务
- `POST /api/restart/:serviceId` - 重启指定的 Gateway 服务
- `GET /api/logs/:serviceId` - 获取指定服务的日志（最近 100 行）

Gateway 管理

- `GET /api/gateways` - 获取所有 Gateway 配置
- `POST /api/gateways` - 创建新的 Gateway
- `PUT /api/gateways/:profileId` - 更新 Gateway 配置
- `DELETE /api/gateways/:profileId` - 删除 Gateway

9.12.8 最佳实践

1. 命名规范

Profile ID：

- 使用小写字母和连字符
- 描述性命名
- 例如: `tech-support`, `content-writer`

Agent ID：

- 与 Profile ID 保持一致
- 添加 `-agent` 后缀
- 例如: `tech-support-agent`

2. 人格设定

清晰明确：

- 使用简洁的语言
- 避免模糊的描述
- 提供具体的例子

保持一致：

- 人格特点要统一
- 回答风格要稳定
- 避免矛盾的设定

定期优化：

- 根据使用反馈调整
- 不断完善人格设定
- 测试不同的配置

3. 安全建议

保护敏感信息：

- 不要在 SOUL.md 中包含密钥
- 定期更换飞书 App Secret
- 限制 Gateway 的网络访问

备份配置：

- 定期备份 `~/.openclaw-*` 目录
- 保存重要的 SOUL.md 文件
- 记录配置变更

9.12.9 项目信息

GitHub 仓库：https://github.com/xianyu110/openclaw-manager

技术栈：

- 前端：React 18 + Tailwind CSS + Vite
- 后端：Express + Node.js
- 状态管理：React Hooks
- 样式：Tailwind CSS

许可证：MIT

作者：Maynor (@xianyu110)

贡献：
欢迎提交 Issue 和 Pull Request！

📝 本章小结

通过本章学习，你已经掌握：

1. **飞书Bot配置**：完整的飞书机器人创建和配置流程
2. **企业微信Bot**：企业微信机器人的配置方法
3. **钉钉Bot配置**：钉钉机器人的接入步骤
4. **QQ Bot配置**：QQ机器人的详细配置

5. **Discord Bot**：Discord机器人的参考配置
6. **平台对比**：各平台的功能对比和选择建议
7. **多Agent配置**：高级的多Agent管理和配置
8. **本地多Agent**：无需绑定IM平台的本地使用
9. **OpenClaw Manager**：可视化管理工具的使用

🎯 实战练习

1. 配置一个飞书机器人并测试基本功能
2. 尝试配置多个Agent，为不同场景使用不同模型
3. 使用 OpenClaw Manager创建和管理多个Gateway
4. 为每个Agent定制专属的人格设定
5. 配置launchd保活服务，实现开机自启

💡 进阶建议

1. 探索更多IM平台的集成方式
2. 优化Agent的人格设定，提升使用体验
3. 使用 OpenClaw Manager简化日常管理
4. 为团队成员创建专属的AI助手
5. 定期备份配置，避免数据丢失

下一章预告：第10章将学习API集成，包括如何对接各种第三方服务，实现更强大的自动化功能。

返回目录：README

飞书Bot配置检查清单

> ✅ 使用本清单确保飞书Bot配置完整，避免常见问题

📋 配置前准备

- [ ] 已安装 OpenClaw（版本 2026.2.9+）
- [ ] 已注册飞书开放平台账号
- [ ] 已创建企业/团队（个人也可以）
- [ ] 网络可以访问飞书开放平台

🔧 应用创建（飞书开放平台）

1. 创建应用

- [ ] 登录飞书开放平台：https://open.feishu.cn
- [ ] 点击"创建企业自建应用"
- [ ] 填写应用信息：
- [ ] 应用名称（如：OpenClaw助手）
- [ ] 应用描述
- [ ] 应用图标

- [ ] 记录 App ID 和 App Secret

2. 配置机器人

- [ ] 进入"应用功能" → "机器人"
- [ ] 启用机器人功能
- [ ] 设置机器人名称
- [ ] 设置机器人描述
- [ ] 上传机器人头像

3. 配置权限（⭐ 重要）

在"权限管理"页面，添加以下权限：

必需权限（缺一不可）

- [ ] `im:message` - 获取与发送单聊、群组消息
- [ ] `im:message:send_as_bot` - 以应用身份发消息

- [ ] `contact:contact.base:readonly` - 获取通讯录基本信息 ⭐

> ⚠️ **特别注意：** `contact:contact.base:readonly` 是最容易遗漏的权限！
>
> 如果缺少此权限：
>
> - ❌ 机器人无法识别用户
> - ❌ 无法响应任何消息
> - ❌ 日志中会出现 "User not found" 错误

可选权限（根据需要添加）

- [ ] `im:message:group_at_msg:readonly` - 获取群组中@机器人的消息
- [ ] `im:message:group_msg:readonly` - 获取群组消息
- [ ] `im:chat` - 获取群组信息
- [ ] `contact:user.base:readonly` - 获取用户详细信息

4. 配置事件订阅

在"事件订阅"页面：

- [ ] 选择"使用长连接接收事件"（WebSocket 模式）
- [ ] 添加事件：`im.message.receive_v1`（接收消息）
- [ ] 确认长连接状态显示"已连接"

> 💡 **提示**：如果长连接无法建立，请先完成下面的 OpenClaw 配置步骤

5. 发布应用

- [ ] 点击"版本管理与发布"
- [ ] 创建版本
- [ ] 填写版本说明
- [ ] 提交审核（企业自建应用通常自动通过）
- [ ] 确认应用状态为"已发布"

6. 设置可用范围

- [ ] 在"可用性"页面设置可用范围
- [ ] 添加可使用的部门或成员
- [ ] 或选择"全员可用"

💻 OpenClaw 配置

1. 添加飞书渠道

```
# 运行添加渠道命令
openclaw channels add

# 选择 Feishu
# 输入 App ID
# 输入 App Secret
```text
**检查配置**：
```bash
# 查看渠道列表
openclaw channels list

# 应该看到 feishu 渠道
```text
### 2. 启动网关

```bash
# 启动网关
openclaw gateway start

# 检查状态
openclaw gateway status

# 应该显示 "running"
```text
### 3. 验证配置

```bash
# 测试飞书连接
openclaw channels test feishu

# 应该显示连接成功
```text
---

## 🧪 测试验证

### 1. 基础测试

- [ ] 在飞书中搜索并添加机器人
- [ ] 发送消息："你好"
- [ ] 机器人应该回复
```
```

```
### 2. 功能测试

- [ ] 测试文本消息
- [ ] 测试文件发送（如果需要）
- [ ] 测试图片发送（如果需要）
- [ ] 测试群组消息（需要@机器人）

### 3. 权限测试

- [ ] 机器人能识别用户名
- [ ] 机器人能正常回复
- [ ] 访问控制正常工作（如果配置了）

---

## 🔍 故障排查

### 机器人不回复？

按以下顺序检查：

1. **检查权限（最常见问题）**
   ```text
   ✅ im:message
   ✅ im:message:send_as_bot
   ✅ contact:contact.base:readonly ⭐ 必需
```

2. **检查事件订阅**

```
✅ 已选择"长连接"模式
✅ 已添加 im.message.receive_v1
✅ 长连接状态"已连接"
```

3. **检查应用状态**

```
✅ 应用已发布
✅ 应用已通过审核
✅ 用户在可用范围内
```

4. **检查 OpenClaw**
```

```
# 检查网关状态
openclaw gateway status

# 查看日志
openclaw logs --follow

# 检查渠道配置
openclaw channels list
```

查看详细日志

```
# 实时查看日志
openclaw logs --follow

# 过滤飞书相关日志
openclaw logs --follow | grep feishu

# 查看最近100行日志
openclaw logs --tail 100
```

常见错误及解决

错误信息	原因	解决方案
`Permission denied`	缺少权限	添加所有必需权限
`User not found`	缺少通讯录权限	添加 `contact:contact.base:readonly`
`Connection failed`	网关未启动	运行 `openclaw gateway start`
`Invalid app_id`	AppID错误	检查配置
`Long connection failed`	长连接失败	确保网关已启动再配置事件订阅

📚 相关文档

- 第9章：多平台集成
- 附录E：常见问题速查
- 飞书开放平台文档

💡 最佳实践

开发建议

1. **先测试后发布**
   - 在测试环境完成配置
   - 充分测试后再发布到生产环境
2. **权限最小化**
   - 只添加必需的权限
   - 定期审查权限使用情况
3. **日志监控**
   - 定期查看日志
   - 及时发现和解决问题
4. **备份配置**
   - 定期备份 OpenClaw 配置
   - 记录飞书应用的配置信息

安全建议

1. **保护密钥**
   - 不要将 App Secret 提交到代码仓库
   - 使用环境变量存储敏感信息
2. **访问控制**
   - 配置 allowlist 限制可访问用户
   - 定期审查访问日志
3. **定期更新**
   - 保持 OpenClaw 版本最新
   - 关注飞书平台的更新公告

最后更新：2026年2月14日
适用版本：OpenClaw 2026.2.9+

第10章 API服务集成（绘图/Notion/视频/语音）

> 接入第三方服务，让AI能力更强大

本章将介绍如何将OpenClaw与各种第三方API服务集成，扩展AI的能力边界。重点介绍4个实用场景的配置方法：AI绘图、Notion数据同步、视频生成和语音合成。

💡 **本章重点**：API集成的配置方法和基础使用。详细的实战案例请参考第14章《创意应用探索》。

10.1 AI绘图服务集成

10.1.1 服务选择

OpenClaw支持多种AI绘图服务：

服务	特点	成本	推荐场景
Gemini图像生成	中文好、白板图擅长	低（0.05-0.4元/张）	教程配图、白板图
DALL-E 3	质量高、风格多	中	创意设计
Midjourney	艺术性强	高	艺术创作
Stable Diffusion	开源免费	免费（需自建）	批量生成

推荐：Gemini图像生成

- ✅ 集成简单（Chat API）
- ✅ 成本低廉
- ✅ 中文支持好
- ✅ 适合教程配图、白板图

10.1.2 安装bananapro-image-gen Skill

方式1：从GitHub安装（推荐）

```
# 克隆技能仓库
git clone https://github.com/xianyu110/awesome-openclaw-tutorial.git

# 复制到OpenClaw的skills目录
cp -r expert-skills-hub/skills/bananapro-image-gen ~/.openclaw/skills/

# 安装依赖
cd ~/.openclaw/skills/bananapro-image-gen
pip3 install -r requirements.txt
```text
**方式2：使用npx命令**

```bash
npx skills add https://github.com/xianyu110/awesome-openclaw-tutorial --skill
bananapro-image-gen
```text
**验证安装：**

```bash
ls ~/.openclaw/skills/bananapro-image-gen
# 应该看到：SKILL.md, README.md, scripts/, test.sh 等文件
```text
### 10.1.3 配置API

**步骤1：获取API 密钥**

1. 访问中转API服务商（推荐支持Gemini的服务）
2. 注册并充值（建议10-20元测试）
3. 获取API Key

**步骤2：配置 OpenClaw**

```bash
nano ~/.openclaw/openclaw.json
```text
添加配置：

```json
{
  "api": {
    "gemini": {
      "apiKey": "your-api-key-here",
```

```
      "baseUrl": "https://apipro.maynor1024.live",
      "model": "gemini-3-pro-image-preview"
    }
  }
}
```text
**步骤3：测试连接**

```bash
cd ~/.openclaw/skills/bananapro-image-gen
bash test.sh
```text
如果看到生成的图片，说明配置成功！

**成本参考：**
- 生成1张1K图片：约0.05-0.1元
- 生成1张2K图片：约0.1-0.2元
- 生成1张4K图片：约0.2-0.4元

💡 **省钱技巧：** 日常使用1K分辨率即可，只有需要打印或高清展示时才用2K/4K。

### 10.1.4 基础使用教程

安装完成后，你可以直接在OpenClaw中使用绘图功能。

**示例1：生成Logo**
```

你：帮我设计1个Logo，主题是"超级个体"，要求：

- 简洁现代
- 使用蓝色和橙色
- 包含一个人和AI的元素
- 适合用在社交媒体头像

```
OpenClaw会自动调用绘图Skill，生成Logo并保存到本地。

**示例2：生成白板图**
```

你：生成一张白板图片，手写字体风格，内容是：

标题：OpenClaw核心优势
```

4大优势：

1. 本地部署 - 保护隐私
2. 文件访问 - 智能搜索
3. Skills扩展 - 无限可能
4. 多平台支持 - 随时随地

用手写字体，添加箭头、框图等手绘元素

```
**示例3：生成社交媒体配图**
```

你：生成一张小红书配图，主题是"AI时代的超级个体"，要求：

- 9:16竖版
- 温暖的配色
- 包含文字：一个人 + AI = 无限可能
- 简约风格

10.1.5 命令行使用（进阶）

如果你想批量生成图片或集成到自动化脚本中，可以直接使用命令行：

基础用法：

```bash
cd ~/.openclaw/skills/bananapro-image-gen

# 生成图片（OpenAI格式）
python3 scripts/generate_image.py \
    --prompt "1个简约的Logo设计，主题是AI助手" \
    --filename output.png \
    --api-format openai

# 生成高清图片（Gemini格式）
python3 scripts/generate_image.py \
    --prompt "一张白板图，手写风格" \
    --filename whiteboard.png \
    --api-format gemini \
    --resolution 2K
```text
**参数说明：**

| 参数 | 说明 | 默认值 | 示例 |
|------|------|--------|------|
| `--prompt` | 图片描述（必填） | - | "1个Logo设计" |
| `--filename` | 输出文件名 | output.png | my_image.png |
| `--api-format` | API格式 | openai | openai/gemini |
| `--resolution` | 分辨率 | 1K | 1K/2K/4K |

**批量生成示例：**

```bash
# 创建批量生成脚本
cat > batch_generate.sh << 'EOF'
#!/bin/bash

# 生成5张不同主题的图片
prompts=(
    "Logo设计：超级个体"
    "白板图：AI工作流"
```

```
    "海报：效率提升10倍"
    "配图：知识管理系统"
    "图标：自动化工具"
)

for i in "${!prompts[@]}"; do
    echo "生成第 $((i+1)) 张图片..."
    python3 scripts/generate_image.py \
        --prompt "${prompts[$i]}" \
        --filename "output_$((i+1)).png"
done

echo "批量生成完成！"
EOF

chmod +x batch_generate.sh
./batch_generate.sh
```text
### 10.1.6 提示词技巧

好的提示词是生成高质量图片的关键。以下是一些实用技巧：

**技巧1：明确风格**
```

❌ 不好：生成一张图片
✅ 好：生成一张白板图片，手写字体风格，简约现代

```
**技巧2：详细描述**
```

❌ 不好：画1个Logo
✅ 好：设计1个Logo，圆形，蓝色渐变，中间是1个抽象的大脑图案，
周围有数据流动的线条，现代科技感

```
**技巧3：指定元素**
```

❌ 不好：画一张架构图
✅ 好：画一张系统架构图，包含：
- 用户端（手机图标）
- 网关（云图标）
```

- 数据库（圆柱图标）
- 用箭头连接各个组件
- 添加文字标注

```
**技巧4：参考风格**
```

✅ 好：生成一张图片，风格类似：
- 手绘白板风格
- 扁平化设计
- 水彩画风格
- 像素艺术风格

提示词模板库：

```bash
# Logo设计模板
"设计1个[主题]Logo，要求：
- 形状：[圆形/方形/抽象]
- 颜色：[主色调]
- 元素：[核心元素]
- 风格：[现代/复古/简约]
- 用途：[社交媒体/网站/名片]"

# 白板图模板
"生成一张白板图片，手写字体风格，内容是：
标题：[标题文字]
核心要点：
1. [要点1]
2. [要点2]
3. [要点3]
用手写字体，添加箭头、框图、下划线等手绘元素"

# 社交媒体配图模板
"生成一张[平台]配图，主题是[主题]，要求：
- 尺寸：[9:16/16:9/1:1]
- 配色：[温暖/冷色/对比]
- 文字：[核心文案]
- 风格：[简约/丰富/创意]"

# 架构图模板
"生成一张[系统名称]架构图，包含：
组件：
- [组件1]：[描述]
- [组件2]：[描述]
- [组件3]：[描述]
连接关系：
- [组件1] → [组件2]
- [组件2] → [组件3]
用框图、箭头、图标等元素，清晰展示系统结构"
```text
### 10.1.7 实战案例

**案例1：为教程生成章节总结图**
```
```

场景：你在写一本OpenClaw教程，需要为每章生成一张总结白板图。

```bash
# 使用提供的测试脚本
cd ~/.openclaw/skills/bananapro-image-gen
bash test_chapters.sh
```text
这个脚本会自动生成4张图片：
- `chapter1_summary.png` - 第1章总结
- `architecture.png` - 核心架构图
- `chapter2_deployment.png` - 部署方式对比
- `chapter3_persona.png` - 人设配置

**效果：**
- 时间节省：手动画图需要30分钟/张，AI生成只需1分钟
- 成本：每张约0.1元
- 质量：手写风格，适合教程配图

**案例2：批量生成社交媒体配图**

场景：你是内容创作者，需要为10篇文章生成配图。

```python
# create_social_images.py
import subprocess
import json

# 文章标题列表
articles = [
    "AI时代的超级个体",
    "效率提升10倍的秘密",
    "一个人的无限可能",
    # ... 更多标题
]

for i, title in enumerate(articles, 1):
    prompt = f"""生成一张小红书配图，主题是"{title}"，要求：
    - 9:16竖版
    - 温暖的配色
    - 包含标题文字
    - 简约现代风格
    """
```

```
    subprocess.run([
        "python3", "scripts/generate_image.py",
        "--prompt", prompt,
        "--filename", f"social_{i}.png",
        "--resolution", "2K"
    ])

    print(f"✅ 生成第{i}张配图")

print("🎉 批量生成完成！")
```text
**效果：**
- 生成10张配图：约10分钟
- 成本：约2元（2K分辨率）
- 手动设计：需要5小时+，成本500元+

**案例3：自动化Logo设计**

场景：你需要为多个项目快速生成Logo方案。

```bash
# 创建Logo生成脚本
cat > generate_logos.sh << 'EOF'
#!/bin/bash

# 项目列表
projects=(
    "超级个体:蓝色:现代科技"
    "知识管理:绿色:简约"
    "效率工具:橙色:活力"
)

for project in "${projects[@]}"; do
    IFS=':' read -r name color style <<< "$project"

    prompt="设计1个${name}的Logo，要求：
    - 主色调：${color}
    - 风格：${style}
    - 简洁现代
    - 适合用在社交媒体"

    python3 scripts/generate_image.py \
        --prompt "$prompt" \
```

```
        --filename "logo_${name}.png"

    echo "✅ 生成 ${name} Logo"
done
EOF

chmod +x generate_logos.sh
./generate_logos.sh
```text
### 10.1.8 常见问题

**Q1：生成的图片不符合预期怎么办？**

A：优化提示词，增加更多细节描述。可以尝试：
- 添加参考风格
- 明确指定元素
- 调整分辨率
- 多生成几次选最好的

**Q2：如何控制成本？**

A：
- 日常使用1K分辨率（约0.05元/张）
- 批量生成时使用脚本，避免重复生成
- 设置每日预算上限
- 使用中转API，成本更低

**Q3：支持哪些图片格式？**

A：目前支持PNG格式，这是最通用的格式。如需其他格式，可以使用转换工具：

```bash
# 转换为JPG
convert output.png output.jpg

# 转换为WebP（更小的文件）
convert output.png output.webp
```text
**Q4：生成速度慢怎么办？**

A：
- 检查网络连接
- 使用国内中转API
```
```

- 降低分辨率（1K最快）
- 避免高峰时段

Q5：如何集成到自动化工作流？

A：参考10.1.5的批量生成示例，或者在OpenClaw的自动化任务中调用：

```python
# 在OpenClaw的自动化脚本中
from skills.nano_banana_image_gen import generate_image

# 生成图片
image_path = generate_image(
    prompt="你的提示词",
    filename="output.png",
    resolution="1K"
)

print(f"图片已生成：{image_path}")
```text
### 10.1.9 进阶技巧

**技巧1：使用两种API格式**

Skill支持两种API格式，可以根据需求切换：

```bash
# OpenAI格式（兼容性好）
python3 scripts/generate_image.py \
    --api-format openai \
    --prompt "你的提示词"

# Gemini格式（功能更强）
python3 scripts/generate_image.py \
    --api-format gemini \
    --prompt "你的提示词" \
    --resolution 2K
```text
**技巧2：图片后处理**

生成图片后，可以使用Python进行后处理：

```python
```

```
from PIL import Image, ImageDraw, ImageFont

# 打开生成的图片
img = Image.open("output.png")

# 添加水印
draw = ImageDraw.Draw(img)
font = ImageFont.truetype("Arial.ttf", 36)
draw.text((10, 10), "© 2026", fill="white", font=font)

# 调整大小
img = img.resize((800, 600))

# 保存
img.save("output_processed.png")
```text
**技巧3：创建图片模板**

```python
# image_templates.py
TEMPLATES = {
    "logo": """设计1个{theme}Logo，要求：
    - 形状：{shape}
    - 颜色：{color}
    - 风格：{style}
    - 用途：{usage}""",

    "whiteboard": """生成一张白板图片，手写字体风格，内容是：
    标题：{title}
    {content}
    用手写字体，添加箭头、框图等手绘元素""",

    "social": """生成一张{platform}配图，主题是{theme}，要求：
    - 尺寸：{ratio}
    - 配色：{color_scheme}
    - 文字：{text}
    - 风格：{style}"""
}

def generate_from_template(template_name, **kwargs):
    prompt = TEMPLATES[template_name].format(**kwargs)
    # 调用生成函数
    return generate_image(prompt)
```

```
# 使用模板
generate_from_template(
    "logo",
    theme="AI助手",
    shape="圆形",
    color="蓝色渐变",
    style="现代科技",
    usage="社交媒体"
)
```text
---

## 10.2 Notion数据同步封装

Notion是流行的知识管理工具，通过OpenClaw集成Notion API，可以实现自动化的数据同步和内容管理。

![Notion工作空间 - 知识管理和协作平台]
(https://upload.maynor1024.live/file/1770869312061_image-20260212120817415.png)

### 10.2.1 为什么要集成Notion？

**应用场景：**
- 📝 自动创建笔记：OpenClaw处理的内容自动保存到Notion
- 📊 数据库同步：自动更新Notion数据库
- 📅 任务管理：自动创建和更新任务
- 🔄 双向同步：Notion和OpenClaw数据互通

**效率提升：**
- 手动复制粘贴：5分钟/次
- 自动同步：5秒/次
- 效率提升：98%

### 10.2.2 Notion集成方案

由于 ClawHub 市场上目前没有现成的 Notion Skills，你可以通过以下方式集成 Notion：

**方案1：使用 Notion API（推荐）**

直接使用 Notion 官方 API 进行集成：

```bash
# 安装 Notion SDK
```

```
npm install @notionhq/client

# 或使用 Python SDK
pip install notion-client
```text
**方案2：自定义 Skill**

创建自己的 Notion Skill：

```bash
# 创建 Skill 目录
mkdir -p ~/.openclaw/skills/my-notion-skill
cd ~/.openclaw/skills/my-notion-skill

# 创建 skill.json
cat > skill.json << 'EOF'
{
  "name": "my-notion-skill",
  "version": "1.0.0",
  "description": "Custom Notion integration",
  "main": "index.js"
}
EOF

# 安装依赖
npm init -y
npm install @notionhq/client
```text
**方案3：使用 MCP 服务器**

如果你使用的是支持 MCP (Model Context Protocol) 的版本，可以配置 Notion MCP 服务器：

```json
{
  "mcpServers": {
    "notion": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-notion"],
      "env": {
        "NOTION_API_KEY": "your-api-key"
      }
    }
  }
```

```
}
```text
💡 **注意**：由于 ClawHub 市场的 Skills 在不断更新，建议：
1. 访问 https://clawhub.com 查看最新可用的 Skills
2. 使用 `npx clawhub@latest search notion` 搜索相关 Skills
3. 如果没有现成的 Skill，可以自己创建或使用 API 直接集成

### 10.2.3 Notion API 完整配置指南

要使用 Notion skill，您需要先进行一些设置。这个技能通过 Notion API 来管理页面、数据库和块。

#### 步骤1：创建 Notion Integration

1. **访问集成页面**
   - 打开浏览器，访问 https://www.notion.so/my-integrations
   - 使用你的 Notion 账号登录

2. **创建新集成**
   - 点击页面右上角的 "+ New integration" 按钮
   - 填写集成信息：
     - **Name**：OpenClaw Integration（或你喜欢的名称）
     - **Associated workspace**：选择你要使用的工作区
     - **Logo**：可选，上传1个图标

3. **配置权限**
   - 在 "Capabilities" 部分，勾选以下权限：
     - ✅ **Read content**：读取页面和数据库内容
     - ✅ **Update content**：更新现有内容
     - ✅ **Insert content**：创建新页面和数据库条目
     - ✅ **Read comments**：读取评论（可选）
     - ✅ **Insert comments**：添加评论（可选）

4. **提交创建**
   - 点击 "Submit" 按钮
   - 系统会生成1个 API 密钥（Secret）

#### 步骤2：获取并保存 API Key

1. **复制 API Key**
   - 在集成详情页面，找到 "Internal Integration Token"
   - 点击 "Show" 按钮显示密钥
   - 点击 "Copy" 按钮复制密钥
   - API 密钥格式通常是：`secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx`
```
```

```
2. **保存 API Key 到配置文件**

   **方式1：使用 OpenClaw 配置文件**
   ```bash
   # 编辑 OpenClaw 配置
   nano ~/.openclaw/openclaw.json
```

添加 Notion 配置：

```
{
  "api": {
    "notion": {
      "apiKey": "secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
      "version": "2022-06-28"
    }
  }
}
```

方式2：使用环境变量

```
# 添加到 ~/.bashrc 或 ~/.zshrc
export NOTION_API_KEY="secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
export NOTION_VERSION="2022-06-28"

# 重新加载配置
source ~/.bashrc  # 或 source ~/.zshrc
```

方式3：使用独立配置文件

```
# 创建 Notion 配置文件
mkdir -p ~/.openclaw/config
cat > ~/.openclaw/config/notion.json << 'EOF'
{
  "apiKey": "secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "version": "2022-06-28"
}
EOF

# 设置文件权限（保护密钥安全）
chmod 600 ~/.openclaw/config/notion.json
```
```

步骤3：分享页面/数据库给 Integration

这是最关键的一步！即使有了 API Key，Integration 也只能访问明确分享给它的页面和数据库。

分享单个页面：

1. 打开你想要 OpenClaw 访问的 Notion 页面
2. 点击页面右上角的 "Share" 按钮
3. 在弹出的对话框中，点击 "Invite" 输入框
4. 搜索你创建的集成名称（例如：OpenClaw Integration）
5. 点击集成名称，然后点击 "Invite"
6. 现在这个页面及其所有子页面都可以被 Integration 访问了

分享数据库：

1. 打开你的 Notion 数据库页面
2. 点击右上角的 "..." 菜单
3. 选择 "Connections" → "Connect to"
4. 搜索并选择你的集成（OpenClaw Integration）
5. 点击 "Confirm"

批量分享（推荐）：

如果你想让 OpenClaw 访问整个工作区：

1. 在工作区的根页面（通常是 "Workspace" 页面）
2. 点击 "Share" → 连接你的 Integration
3. 这样所有子页面和数据库都会被自动分享

步骤4：获取 Database ID（如需操作数据库）

如果你要让 OpenClaw 操作特定的数据库，需要获取 Database ID。

方法1：从 URL 获取

1. 在浏览器中打开 Notion 数据库页面
2. 复制浏览器地址栏的 URL
3. Database ID 是 URL 中的 32 位字符串

示例 URL：

```
https://www.notion.so/workspace/DatabaseName-1234567890abcdef1234567890abcdef?
v=...
                                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                                                    这部分就是 Database ID
```text
去掉连字符后的格式：
```
```

1234567890abcdef1234567890abcdef

```
**方法2：使用 Notion API 查询**

```bash
# 使用 curl 查询所有可访问的数据库
curl -X POST https://api.notion.com/v1/search \
  -H "Authorization: Bearer secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Notion-Version: 2022-06-28" \
  -H "Content-Type: application/json" \
  -d '{
    "filter": {
      "property": "object",
      "value": "database"
    }
  }'
```text
**保存 Database ID：**

```bash
# 添加到 OpenClaw 配置
nano ~/.openclaw/openclaw.json
```text
```json
{
  "api": {
    "notion": {
      "apiKey": "secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx",
      "version": "2022-06-28"
    }
  },
  "notion": {
    "databases": {
      "notes": "1234567890abcdef1234567890abcdef",
      "tasks": "abcdef1234567890abcdef1234567890",
      "projects": "567890abcdef1234567890abcdef1234"
    },
    "defaultDatabase": "notes"
  }
}
```
```

步骤5：测试连接

测试1：使用 curl 测试

```
# 设置变量
NOTION_API_KEY="secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
NOTION_VERSION="2022-06-28"

# 测试 API 连接
curl -X GET https://api.notion.com/v1/users/me \
  -H "Authorization: Bearer $NOTION_API_KEY" \
  -H "Notion-Version: $NOTION_VERSION"

# 如果成功，会返回你的用户信息
```text
**测试2：查询数据库**

```bash
# 查询特定数据库
DATABASE_ID="1234567890abcdef1234567890abcdef"

curl -X POST "https://api.notion.com/v1/databases/$DATABASE_ID/query" \
  -H "Authorization: Bearer $NOTION_API_KEY" \
  -H "Notion-Version: $NOTION_VERSION" \
  -H "Content-Type: application/json" \
  -d '{}'

# 如果成功，会返回数据库中的条目
```text
**测试3：创建页面**

```bash
# 在数据库中创建1个测试页面
curl -X POST https://api.notion.com/v1/pages \
  -H "Authorization: Bearer $NOTION_API_KEY" \
  -H "Notion-Version: $NOTION_VERSION" \
  -H "Content-Type: application/json" \
  -d '{
    "parent": {
      "database_id": "'$DATABASE_ID'"
    },
    "properties": {
      "Name": {
        "title": [
          {
            "text": {
```

```
              "content": "OpenClaw 测试页面"
            }
          }
        ]
      }
    }
  }'

# 如果成功，会返回新创建页面的信息
```text
**测试4：使用 OpenClaw 测试**

```bash
# 如果安装了 Notion Skill
openclaw notion test

# 或者直接在 OpenClaw 中测试
你：测试 Notion 连接

OpenClaw：正在测试 Notion API 连接...

✅ API 连接成功！

📊 连接信息：
• API 版本：2022-06-28
• 用户名：Your Name
• 工作区：Your Workspace
• 可访问页面：15个
• 可访问数据库：3个

📁 可用数据库：
1. 笔记数据库 (notes)
2. 任务数据库 (tasks)
3. 项目数据库 (projects)

✅ Notion 集成配置完成！
```text
#### 常见问题排查

**问题1：API 调用返回 401 Unauthorized**

原因：API Key 错误或未正确配置
```
```

```
解决方案：
```bash
# 检查 API Key 格式
echo $NOTION_API_KEY
# 应该是：secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# 重新复制 API Key
# 访问 https://www.notion.so/my-integrations
# 重新复制密钥并更新配置
```text
**问题2：API 调用返回 404 Not Found**

原因：页面或数据库未分享给 Integration

解决方案：
1. 确认页面/数据库已分享给 Integration
2. 检查 Database ID 是否正确
3. 确认 Integration 有访问权限

**问题3：无法创建页面**

原因：Integration 缺少 "Insert content" 权限

解决方案：
1. 访问 https://www.notion.so/my-integrations
2. 选择你的 Integration
3. 在 "Capabilities" 中勾选 "Insert content"
4. 保存更改

**问题4：Database ID 找不到**

解决方案：
```bash
# 使用 API 搜索所有数据库
curl -X POST https://api.notion.com/v1/search \
  -H "Authorization: Bearer $NOTION_API_KEY" \
  -H "Notion-Version: $NOTION_VERSION" \
  -H "Content-Type: application/json" \
  -d '{
    "filter": {
      "property": "object",
      "value": "database"
    }
```
```

```
  }' | jq '.results[] | {id: .id, title: .title}'
```text
#### 安全建议

**1. 保护 API Key**
```bash
# 设置配置文件权限
chmod 600 ~/.openclaw/config/notion.json

# 不要将 API Key 提交到 Git
echo "*.json" >> ~/.openclaw/.gitignore
echo "openclaw.json" >> ~/.gitignore
```text
**2. 使用环境变量**
```bash
# 在 ~/.bashrc 或 ~/.zshrc 中设置
export NOTION_API_KEY="secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx"

# 在脚本中使用
NOTION_API_KEY="${NOTION_API_KEY}"
```text
**3. 定期轮换密钥**
```bash
# 每3-6个月更换一次 API Key
# 1. 在 Notion 中创建新的 Integration
# 2. 更新配置文件
# 3. 删除旧的 Integration
```text
**4. 最小权限原则**
- 只授予必要的权限
- 只分享需要访问的页面/数据库
- 定期审查 Integration 的访问权限

#### 完整配置示例

```json
{
  "api": {
    "notion": {
      "apiKey": "secret_xxxxxxxxxxxxxxxxxxxxxxxxxxxx",
      "version": "2022-06-28",
      "timeout": 30000,
      "retries": 3
```

```
    }
  },
  "notion": {
    "databases": {
      "notes": "1234567890abcdef1234567890abcdef",
      "tasks": "abcdef1234567890abcdef1234567890",
      "projects": "567890abcdef1234567890abcdef1234",
      "meetings": "890abcdef1234567890abcdef1234567"
    },
    "defaultDatabase": "notes",
    "autoSync": true,
    "syncInterval": 300,
    "templates": {
      "note": {
        "properties": {
          "Tags": ["学习笔记"],
          "Status": "草稿"
        }
      },
      "task": {
        "properties": {
          "Status": "待办",
          "Priority": "中"
        }
      }
    }
  }
}
```text
完成这些设置后，你就可以通过 OpenClaw 调用 Notion API 来创建、读取和更新 Notion 中的内容了！

### 10.2.4 配置Notion集成

```bash
# 编辑OpenClaw 配置
nano ~/.openclaw/openclaw.json
```text
添加Notion配置：

```json
{
  "api": {
    "notion": {
```

```
      "apiKey": "secret_xxxxxxxxxxxxx",
      "version": "2022-06-28"
    }
  },
  "notion": {
    "defaultDatabase": "your-database-id",
    "autoSync": true,
    "syncInterval": 300
  }
}
```text
**配置说明：**

| 参数 | 说明 | 示例 |
|------|------|------|
| apiKey | Notion API 密钥 | secret_xxx |
| version | API版本 | 2022-06-28 |
| defaultDatabase | 默认数据库ID | 32位字符串 |
| autoSync | 自动同步 | true/false |
| syncInterval | 同步间隔（秒） | 300 |

### 10.2.5 基础使用与实战案例

**示例1：自动创建笔记**
```

你：把这篇文章保存到Notion

标题：OpenClaw使用心得
内容：
今天学习了OpenClaw的API集成功能，
主要收获有：

1. 学会了配置Gemini API
2. 成功生成了第一张AI图片
3. 了解了Skills扩展机制

OpenClaw：好的，正在保存到Notion...

📝 【创建Notion页面】

✅ 页面创建成功！

📄 页面信息：
- 标题：OpenClaw使用心得
```

- 创建时间：2026-02-12 10:30
- 页面ID：abc123...
- 链接：https://notion.so/abc123

📊 内容统计：
- 字数：120字
- 段落：4段
- 列表项：3项

已自动添加：
- 标签：#学习笔记 #OpenClaw
- 分类：技术文档
- 创建日期：2026-02-12

🔗 在Notion中打开：
https://notion.so/abc123

```
**示例2：批量导入文档**
```

你：把这个文件夹里的所有Markdown文件导入到Notion

文件夹：~/Documents/notes/

OpenClaw：好的，正在扫描文件...

📁 【批量导入Notion】

📊 扫描结果：
- 文件夹：~/Documents/notes/
- 文件数量：15个
- 总大小：2.5MB
- 预计时间：2分钟

⏳ 导入进度：

[████████████████████] 100% (15/15)

✅ 导入完成！

📊 导入统计：
- 成功：15个
- 失败：0个
- 总耗时：1分45秒

📁 Notion结构：
笔记导入/
├── 学习笔记/

```
│   ├── 学习笔记.md
│   └── 技术文档.md
├── 项目管理/
│   ├── 项目规划.md
│   └── 会议记录.md
└── 产品设计/
    └── 产品需求.md
```

🔗 在Notion中查看：
https://notion.so/workspace/notes-import

10.2.6 进阶技巧

技巧1：自定义模板

```bash
# 创建会议记录模板
openclaw notion template create "meeting" \
  --title "会议记录-{date}" \
  --properties "主题,时间,参会人,内容,决策,任务" \
  --database "meetings-db-id"

# 使用模板
你：用会议模板创建今天的会议记录
```text
**技巧2：自动化工作流**

```bash
# 设置自动同步规则
openclaw config set notion.auto-rules '{
  "markdown-to-notion": {
    "watch": "~/Documents/notes/",
    "target": "notes-db-id",
    "trigger": "file-save"
  },
  "task-sync": {
    "watch": "~/Projects/tasks.md",
    "target": "tasks-db-id",
    "trigger": "file-change"
  }
}'
```text
**技巧3：双向同步**

```bash
# 启用双向同步
openclaw config set notion.bidirectional true

# Notion更新会自动同步到本地
# 本地更新会自动同步到Notion
```text
### 10.2.7 常见问题

**Q1：API调用失败怎么办？**

A：检查以下几点：
- API Key是否正确
- Integration是否已分享给页面
- API版本是否匹配
- 网络连接是否正常

```bash
# 测试API连接
openclaw notion test

# 查看错误日志
openclaw logs --filter notion
```text
**Q2：如何处理大量数据？**

A：使用批量操作和分页：

```bash
# 批量导入（自动分批）
openclaw notion import ~/notes/ \
  --batch-size 10 \
  --delay 1000

# 分页查询
openclaw notion query \
  --database "db-id" \
  --page-size 100
```text
**Q3：如何避免重复创建？**

A：使用唯一标识符：

```bash
# 配置去重规则
openclaw config set notion.dedup '{
  "enabled": true,
  "field": "title",
  "action": "skip"
}'
```text
### 10.2.8 效率提升数据

| 任务类型 | 手动操作 | 自动化 | 节省时间 | 提升比例 |
|---|---|---|---|---|
| 创建笔记 | 5分钟 | 5秒 | 4分55秒 | 98.3% |
| 更新数据库 | 10分钟 | 10秒 | 9分50秒 | 98.3% |
| 批量导入 | 60分钟 | 2分钟 | 58分钟 | 96.7% |
| 会议记录 | 30分钟 | 1分钟 | 29分钟 | 96.7% |

**成本参考：**
- Notion API：免费（有速率限制）
- OpenClaw Skills：免费
- 总成本：$0/月

---

## 10.3 视频生成服务

AI视频生成是新兴的创意工具，可以将文字脚本自动转换为视频。

### 10.3.1 为什么要集成视频服务？

**应用场景：**
- 🎬 短视频创作：快速生成抖音、快手短视频
- 📺 长视频制作：B站、YouTube教程视频
- 🎥 营销视频：产品介绍、广告宣传
- 🎞️ 教育视频：课程讲解、知识分享

**效率提升：**
- 手动剪辑：2-4小时/视频
- AI生成：10-30分钟/视频
- 效率提升：80-90%

### 10.3.2 推荐的视频生成Skills

根据ClawHub技能市场，以下是推荐的视频生成Skills：

**1. video-agent - HeyGen视频生成**
```bash
# 安装
npx clawhub@latest install video-agent

# 功能
• 使用HeyGen API生成AI头像视频
```

```
• 支持多种语言和音色
• 专业级视频质量
• 适合教程和营销视频
```text
**2. sora-video-gen - OpenAI Sora**
```bash
# 安装
npx clawhub@latest install sora-video-gen

# 功能
• 使用OpenAI Sora API
• 文本生成视频
• 高质量视频输出
• 创意视频制作
```text
**3. veo3-video-gen - Google Veo**
```bash
# 安装
npx clawhub@latest install veo3-video-gen

# 功能
• 使用Google Veo 3.x
• 生成和拼接短视频
• 支持多种风格
• 适合短视频创作
```text
**4. tube-cog - YouTube内容创作**
```bash
# 安装
npx clawhub@latest install tube-cog

# 功能
• YouTube内容创作
• 由CellCog提供支持
• 完整视频制作流程
• 自动化发布
```text
**5. video-cog - 长篇视频制作**
```bash
# 安装
npx clawhub@latest install video-cog

# 功能
```

```
• 长篇AI视频制作
• 多智能体协作
• 专业级视频输出
• 适合教程和课程
```text
### 10.3.3 配置视频服务

**配置HeyGen（推荐）：**

```bash
# 编辑OpenClaw 配置
nano ~/.openclaw/openclaw.json
```text
添加HeyGen配置：

```json
{
  "api": {
    "heygen": {
      "apiKey": "your-heygen-api-key",
      "baseUrl": "https://api.heygen.com/v1"
    }
  },
  "video": {
    "defaultAvatar": "avatar-id",
    "defaultVoice": "voice-id",
    "quality": "high",
    "format": "mp4"
  }
}
```text
**获取HeyGen API Key：**
1. 访问 https://www.heygen.com
2. 注册并登录账号
3. 进入 API Settings
4. 创建并复制 API Key

**成本参考：**
- HeyGen：$0.5-2/分钟（根据质量）
- Sora：$0.1-0.5/秒
- Veo：$0.05-0.2/秒

### 10.3.4 基础使用
```
```

```
**示例1：生成AI头像视频**
```

你：生成1个30秒的AI头像视频，内容是：

主题：OpenClaw快速入门
脚本：
您好，我是OpenClaw AI助手。
今天介绍如何快速上手OpenClaw。
只需3步：安装、配置、使用。
开始吧！

要求：

- 头像：专业男性
- 音色：温和
- 背景：科技感
- 字幕：中文

OpenClaw：好的，正在生成视频...

🎬 【AI视频生成】OpenClaw快速入门

📊 视频信息：
• 时长：30秒
• 分辨率：1080p
• 格式：MP4
• 大小：约15MB

⏳ 生成进度：

1. 脚本分析 [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
2. 头像渲染 [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
3. 语音合成 [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
4. 视频合成 [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
5. 字幕添加 [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%

✅ 视频生成完成！

📹 视频预览：
[视频缩略图]

📊 视频详情：
• 头像：专业男性（ID: avatar-001）
• 音色：温和男声（ID: voice-cn-male-01）

• 背景：科技蓝色渐变
• 字幕：中文，白色，底部居中

💰 成本：
• 生成费用：1.00•存储费用：0.01
• 总计：$1.01

📁 文件已保存：
• 视频文件：openclaw_intro.mp4
• 字幕文件：openclaw_intro.srt
• 脚本文件：openclaw_intro.txt

🔗 在线预览：
https://video.openclaw.com/abc123

```
**示例2：批量生成系列视频**
```

你：批量生成OpenClaw教程系列视频，共5集

第1集：什么是OpenClaw
第2集：如何安装 OpenClaw
第3集：基础配置教程
第4集：Skills扩展使用
第5集：实战案例分享

OpenClaw：好的，正在批量生成...

🎬 【批量视频生成】OpenClaw教程系列

📊 任务信息：
• 视频数量：5集
• 每集时长：3-5分钟
• 总时长：约20分钟
• 预计时间：15分钟

⏳ 生成进度：

第1集：什么是OpenClaw
[████████████████████] 100% ✅

第2集：如何安装 OpenClaw
[████████████████████] 100% ✅

第3集：基础配置教程
[████████████████████] 100% ✅

第4集：Skills扩展使用
[████████████████████] 100% ✅

第5集：实战案例分享
[████████████████████] 100% ✅

✅ 全部生成完成！

📊 生成统计：
- 成功：5集
- 失败：0集
- 总耗时：12分30秒
- 总大小：150MB

📁 文件列表：

1. ✅ 01_什么是OpenClaw.mp4 (30MB)
2. ✅ 02_如何安装 OpenClaw.mp4 (28MB)
3. ✅ 03_基础配置教程.mp4 (32MB)
4. ✅ 04_Skills扩展使用.mp4 (35MB)
5. ✅ 05_实战案例分享.mp4 (25MB)

💰 总成本：
- 生成费用：15.00•存储费用：0.15
- 总计：$15.15

📦 已打包：
- OpenClaw教程系列.zip (150MB)

🔗 下载链接：
https://video.openclaw.com/series/abc123

```
### 10.3.5 实战案例

**案例1：自动化短视频创作**

场景：每天自动生成一条知识分享短视频。

```bash
# 创建自动化脚本
cat > daily_video.sh << 'EOF'
#!/bin/bash

# 获取今日主题
topic=$(openclaw ask "推荐1个适合短视频的技术主题")

# 生成脚本
script=$(openclaw ask "为主题'${topic}'生成60秒短视频脚本")

# 生成视频
openclaw video generate \
  --script "$script" \
  --duration 60 \
  --avatar "professional-male" \
  --voice "cn-male-01" \
  --output "daily_$(date +%Y%m%d).mp4"

echo "✅ 今日视频生成完成：${topic}"
EOF

chmod +x daily_video.sh

# 设置定时任务（每天早上9点）
openclaw schedule add "daily-video" \
  --time "09:00" \
  --command "./daily_video.sh"
```text
**案例2：产品介绍视频生成**

场景：为新产品快速生成多语言介绍视频。
```

你：为OpenClaw生成产品介绍视频，需要中英日3个版本

产品特点：

1. 本地部署，保护隐私
2. 文件访问，智能搜索
3. Skills扩展，无限可能
4. 多平台支持，随时随地

OpenClaw：好的，正在生成多语言版本...

🎬 【多语言视频生成】

📊 任务信息：
- 语言版本：3个（中文、英文、日文）
- 视频时长：每个2分钟
- 预计时间：10分钟

⏳ 生成进度：

中文版本：
[████████████████████] 100% ✅

英文版本：
[████████████████████] 100% ✅

日文版本：
[████████████████████] 100% ✅

✅ 全部生成完成！

📁 文件列表：

1. ✅ OpenClaw_Intro_CN.mp4 (40MB)
2. ✅ OpenClaw_Intro_EN.mp4 (38MB)
3. ✅ OpenClaw_Intro_JP.mp4 (39MB)

💰 总成本：$6.00

🔗 在线预览：
- 中文：https://video.openclaw.com/cn/abc123
- 英文：https://video.openclaw.com/en/abc123
- 日文：https://video.openclaw.com/jp/abc123

10.3.6 进阶技巧

技巧1：自定义头像

```bash
# 上传自定义头像照片
openclaw video avatar upload \
  --photo "my_photo.jpg" \
  --name "my-avatar"

# 使用自定义头像
openclaw video generate \
  --avatar "my-avatar" \
  --script "你的脚本"
```text
**技巧2：视频后期处理**

```bash
# 添加片头片尾
openclaw video edit \
  --input "video.mp4" \
  --intro "intro.mp4" \
  --outro "outro.mp4" \
  --output "final.mp4"

# 添加背景音乐
openclaw video edit \
  --input "video.mp4" \
  --bgm "music.mp3" \
  --volume 0.3 \
  --output "with_music.mp4"
```text
**技巧3：批量处理优化**

```bash
# 并行生成多个视频
openclaw video batch \
  --scripts "scripts/*.txt" \
  --parallel 3 \
  --output "videos/"
```text
### 10.3.7 常见问题

**Q1：视频生成失败怎么办？**

A：检查以下几点：
- API Key是否正确
- 账户余额是否充足
- 脚本是否符合要求
- 网络连接是否稳定

**Q2：如何提高视频质量？**

A：
- 使用高质量模式
- 优化脚本内容
- 选择合适的头像和音色
- 添加背景音乐和特效

**Q3：如何控制成本？**

A：
- 使用标准质量模式
- 批量生成享受折扣
- 复用头像和音色
- 合理规划视频时长

### 10.3.8 效率提升数据

| 任务类型 | 手动制作 | AI生成 | 节省时间 | 提升比例 |
|---|---|---|---|---|
| 短视频（1分钟） | 120分钟 | 10分钟 | 110分钟 | 91.7% |
| 中视频（5分钟） | 300分钟 | 30分钟 | 270分钟 | 90.0% |
| 长视频（15分钟） | 600分钟 | 60分钟 | 540分钟 | 90.0% |
| 批量生成（5个） | 1500分钟 | 120分钟 | 1380分钟 | 92.0% |

**成本对比：**
- 人工制作：$50-200/分钟
- AI生成：$0.5-2/分钟
- 成本节省：95-99%

---

## 10.4 语音合成接入

将文字转换为语音，适用于有声读物、视频配音、语音助手等场景。

### 10.4.1 为什么要集成语音服务？

**应用场景：**
- 📚 有声读物：将文章、书籍转换为音频
- 🎙️ 视频配音：为视频添加专业配音
- 🔊 语音助手：实现语音交互功能
- 📻 播客制作：快速生成播客内容

**效率提升：**
- 人工录音：30-60分钟/千字
- AI合成：1-2分钟/千字
- 效率提升：95-98%

### 10.4.2 推荐的语音合成Skills

根据ClawHub技能市场，以下是推荐的语音合成Skills：

**1. elevenlabs - 顶级语音合成**
```bash
# 安装
npx clawhub@latest install elevenlabs

# 功能
• 超自然的语音质量
• 支持多种语言和音色
• 情感表达丰富
• 适合专业配音
```text
**2. azure-tts - 微软语音服务**
```bash
# 安装
npx clawhub@latest install azure-tts

# 功能
• 支持140+语言
• 神经网络语音
• 高质量输出
• 企业级稳定性
```text
**3. google-tts - Google语音**
```bash
```

```
# 安装
npx clawhub@latest install google-tts

# 功能
• WaveNet技术
• 自然流畅
• 多语言支持
• 性价比高
```text
**4. openai-tts - OpenAI语音**
```bash
# 安装
npx clawhub@latest install openai-tts

# 功能
• 使用OpenAI TTS API
• 6种高质量音色
• 支持多种语言
• 简单易用
```text
### 10.4.3 支持的语音服务对比

| 服务 | 特点 | 成本 | 音质 | 语言支持 |
|------|------|------|------|----------|
| ElevenLabs | 最自然 | 高 | ⭐⭐⭐⭐⭐ | 29种 |
| Azure TTS | 企业级 | 中 | ⭐⭐⭐⭐⭐ | 140+种 |
| Google TTS | 性价比高 | 中 | ⭐⭐⭐⭐ | 100+种 |
| OpenAI TTS | 简单易用 | 低 | ⭐⭐⭐⭐ | 50+种 |
| 讯飞语音 | 中文好 | 低 | ⭐⭐⭐⭐ | 中文为主 |

### 10.4.4 配置语音服务

**配置ElevenLabs（推荐）：**

```bash
# 编辑OpenClaw 配置
nano ~/.openclaw/openclaw.json
```text
添加ElevenLabs配置：

```json
{
  "api": {
```

```
    "elevenlabs": {
      "apiKey": "your-elevenlabs-api-key",
      "baseUrl": "https://api.elevenlabs.io/v1"
    }
  },
  "tts": {
    "defaultVoice": "voice-id",
    "defaultModel": "eleven_multilingual_v2",
    "quality": "high",
    "format": "mp3"
  }
}
```text
**获取ElevenLabs API Key：**
1. 访问 https://elevenlabs.io
2. 注册并登录账号
3. 进入 Profile → API Keys
4. 创建并复制 API Key

**成本参考：**
- ElevenLabs：$0.30/千字符
- Azure TTS：$0.016/千字符
- Google TTS：$0.016/千字符
- OpenAI TTS：$0.015/千字符

### 10.4.5 基础使用

**示例1：文本转语音**
```

你：把这段文字转换为语音：

OpenClaw是1个开源的AI助手框架，
它可以访问本地文件，操作系统应用，
扩展各种功能，让AI真正成为你的助手。

要求：

- 音色：专业男声
- 语速：正常
- 格式：MP3
- 质量：高清

OpenClaw：好的，正在转换...
```

🔊 【文本转语音】

📝 文本信息：
• 字数：52字
• 语言：中文
• 预计时长：约15秒

⏳ 转换进度：

1. 文本分析 [████████████████████] 100%
2. 语音合成 [████████████████████] 100%
3. 音频处理 [████████████████████] 100%
4. 格式转换 [████████████████████] 100%

✅ 转换完成！

🎵 音频信息：
• 时长：15秒
• 格式：MP3
• 比特率：192kbps
• 大小：360KB

📊 音频详情：
• 音色：专业男声（ID: voice-cn-male-01）
• 语速：正常（1.0x）
• 音量：标准（0dB）
• 质量：高清

💰 成本：
• 转换费用：0.02•存储费用：0.001
• 总计：$0.021

📁 文件已保存：
• 音频文件：openclaw_intro.mp3
• 文本文件：openclaw_intro.txt

🔗 在线试听：
https://audio.openclaw.com/abc123

```
**示例2：批量转换文章**
```

你：把这个文件夹里的所有文章都转换为音频

文件夹：~/Documents/articles/

OpenClaw：好的，正在扫描文件...

🔊 【批量文本转语音】

📊 扫描结果：
• 文件夹：~/Documents/articles/
• 文件数量：10篇
• 总字数：25,000字
• 预计时长：约50分钟
• 预计时间：5分钟

⏳ 转换进度：

[████████████████████] 100% (10/10)

✅ 全部转换完成！

📊 转换统计：
• 成功：10篇
• 失败：0篇
• 总耗时：4分30秒
• 总时长：48分钟
• 总大小：115MB

📁 文件列表：

1. ✅ AI时代的超级个体.mp3 (12MB, 5分钟)
2. ✅ 效率提升10倍的秘密.mp3 (10MB, 4分钟)
3. ✅ 一个人的无限可能.mp3 (15MB, 6分钟)
4. ✅ 知识管理系统搭建.mp3 (11MB, 4.5分钟)
5. ✅ 自动化工作流设计.mp3 (13MB, 5.5分钟)
   ... (共10个)

💰 总成本：
• 转换费用：7.50•存储费用：0.12
• 总计：$7.62

📦 已打包：
• 有声文章合集.zip (115MB)

🔗 下载链接：
https://audio.openclaw.com/batch/abc123

```
### 10.4.6 实战案例

**案例1：自动化有声读物制作**

场景：将电子书自动转换为有声读物。
```

你：把这本电子书转换为有声读物

文件：OpenClaw完全指南.pdf
要求：

- 分章节转换
- 每章单独文件
- 添加章节标题语音
- 生成播放列表

OpenClaw：好的，正在处理...

📚 【有声读物制作】

📖 电子书信息：
・书名：OpenClaw完全指南
・章节数：16章
・总字数：80,000字
・预计时长：约3小时

⏳ 处理进度：

1. PDF文本提取 [████████████████████] 100%
2. 章节分割 [████████████████████] 100%
3. 语音合成 [████████████████████] 100%
4. 音频处理 [████████████████████] 100%
5. 播放列表生成 [████████████████████] 100%

✅ 制作完成！

📁 文件结构：
OpenClaw完全指南_有声版/
├── 00_书名和前言.mp3 (5分钟)
├── 01_第1章_OpenClaw简介.mp3 (10分钟)
├── 02_第2章_快速开始.mp3 (12分钟)
├── 03_第3章_基础配置.mp3 (15分钟)
... (共17个文件)

├──── playlist.m3u (播放列表)
└──── metadata.json (元数据)

📊 制作统计：
- 总时长：3小时5分钟
- 总大小：450MB
- 音频格式：MP3 192kbps
- 章节数：16章 + 前言

💰 总成本：$24.00

📦 已打包：
- OpenClaw完全指南_有声版.zip (450MB)

🔗 下载链接：
https://audio.openclaw.com/audiobook/abc123

```
**案例2：多语言配音生成**

场景：为产品介绍视频生成多语言配音。
```

你：为这个视频脚本生成中英日三种语言的配音

脚本：
OpenClaw是1个开源的AI助手框架。
它可以访问本地文件，操作系统应用。
让AI真正成为你的超级助手。

OpenClaw：好的，正在生成多语言配音...

🎙️ 【多语言配音生成】

📊 任务信息：
- 语言版本：3个（中文、英文、日文）
- 脚本字数：52字
- 预计时长：每个约15秒

⏳ 生成进度：

中文配音：
[████████████████████] 100% ✅

英文配音：
[████████████████████] 100% ✅

日文配音：
[████████████████████] 100% ✅

✅ 全部生成完成！

📁 文件列表：

1. ✅ openclaw_intro_cn.mp3 (360KB, 15秒)
2. ✅ openclaw_intro_en.mp3 (380KB, 16秒)
3. ✅ openclaw_intro_jp.mp3 (370KB, 15秒)

📊 配音详情：
- 中文：专业男声，语速正常
- 英文：美式男声，语速正常
- 日文：标准男声，语速正常

💰 总成本：$0.06

🔗 在线试听：
- 中文：https://audio.openclaw.com/cn/abc123
- 英文：https://audio.openclaw.com/en/abc123
- 日文：https://audio.openclaw.com/jp/abc123

10.4.7 进阶技巧

技巧1：自定义音色

```bash
# 克隆自己的声音（ElevenLabs）
openclaw tts voice clone \
  --samples "voice_samples/*.mp3" \
  --name "my-voice"

# 使用自定义音色
openclaw tts generate \
  --text "你的文本" \
  --voice "my-voice" \
  --output "output.mp3"
```text
**技巧2：情感控制**

```bash
# 添加情感标记
openclaw tts generate \
  --text "这真是太棒了！[excited]" \
  --voice "voice-id" \
  --emotion "excited" \
  --output "excited.mp3"
```text
**技巧3：语速和音调调整**

```bash
# 调整语速和音调
openclaw tts generate \
  --text "你的文本" \
  --voice "voice-id" \
  --speed 1.2 \
  --pitch 1.1 \
  --output "adjusted.mp3"
```text
**技巧4：批量处理优化**

```bash
# 并行转换多个文件
openclaw tts batch \
```

```
  --input "texts/*.txt" \
  --voice "voice-id" \
  --parallel 5 \
  --output "audios/"
```text
### 10.4.8 常见问题

**Q1：语音听起来不自然怎么办？**

A：
- 使用更高质量的模型
- 优化文本标点符号
- 调整语速和音调
- 尝试不同的音色

**Q2：如何处理长文本？**

A：
- 自动分段处理
- 使用流式合成
- 批量转换后合并

```bash
# 长文本自动分段
openclaw tts generate \
  --text-file "long_article.txt" \
  --auto-split true \
  --max-length 5000 \
  --output "output.mp3"
```text
**Q3：如何控制成本？**

A：
- 选择性价比高的服务
- 批量处理享受折扣
- 复用常用音色
- 合理控制音频质量

**Q4：支持哪些音频格式？**

A：支持多种格式：
- MP3：通用格式，文件小
- WAV：无损格式，质量高
```
```

- OGG：开源格式，压缩好
- AAC：高质量，文件小

```bash
# 指定输出格式
openclaw tts generate \
  --text "你的文本" \
  --format "wav" \
  --output "output.wav"
```text
### 10.4.9 效率提升数据

| 任务类型 | 人工录音 | AI合成 | 节省时间 | 提升比例 |
|---------|---------|--------|----------|----------|
| 短文本（100字） | 10分钟 | 30秒 | 9.5分钟 | 95.0% |
| 中文本（1000字） | 60分钟 | 2分钟 | 58分钟 | 96.7% |
| 长文本（10000字） | 600分钟 | 15分钟 | 585分钟 | 97.5% |
| 批量转换（10篇） | 6000分钟 | 120分钟 | 5880分钟 | 98.0% |

**成本对比：**
- 人工录音：$50-200/千字
- AI合成：$0.015-0.30/千字
- 成本节省：99%+

**质量对比：**
- 人工录音：⭐⭐⭐⭐⭐（最自然，但成本高）
- AI合成（高端）：⭐⭐⭐⭐⭐（接近人工，成本低）
- AI合成（标准）：⭐⭐⭐⭐（质量好，性价比高）

---

## 📦 本章Skills完整安装指南

### ⚠️ 重要说明

在安装 Skills 之前，请注意：

1. **验证 Skill 是否存在**
   ```bash
   # 搜索 Skill
   npx clawhub@latest search <skill-name>
   ```

```
# 查看 Skill 详情
npx clawhub@latest info <skill-name>
```

2. **ClawHub 市场动态变化**
   - Skills 可能被移除、更名或更新
   - 本文档中的某些 Skills 可能不存在
   - 建议访问 https://clawhub.com 查看最新列表
3. **替代方案**
   - 如果 Skill 不存在，可以使用服务的官方 API
   - 可以自己创建自定义 Skill
   - 可以在 GitHub 搜索社区贡献的 Skills
4. **安全提示**
   - 只安装来自可信来源的 Skills
   - 查看 Skill 的源代码和评价
   - 注意保护 API 密钥等敏感信息

安装所有API集成Skills

方式1：使用提供的批量安装脚本（推荐）

```
# 下载并运行批量安装脚本
curl -O https://raw.githubusercontent.com/xianyu110/awesome-openclaw-
tutorial/main/scripts/install_api_skills.sh
chmod +x install_api_skills.sh

# 查看使用帮助
./install_api_skills.sh --help

# 安装特定场景的Skills
./install_api_skills.sh content    # 内容创作者
./install_api_skills.sh video      # 视频UP主
./install_api_skills.sh knowledge  # 知识工作者
./install_api_skills.sh marketing  # 营销人员
./install_api_skills.sh education  # 教育工作者

# 安装所有Skills
./install_api_skills.sh all
```text
**方式2：手动逐个安装**

```bash
# AI绘图服务 Skills（需要先验证是否存在）
# 使用前建议先搜索：npx clawhub@latest search <skill-name>
npx clawhub@latest install fal-ai
npx clawhub@latest install nvidia-image-gen
npx clawhub@latest install pollinations
npx clawhub@latest install venice-ai
npx clawhub@latest install recraft

# Notion集成（目前ClawHub可能没有现成的Skill）
# 建议使用 Notion API 直接集成或自定义 Skill

# 视频生成服务 Skills
npx clawhub@latest install video-agent
npx clawhub@latest install sora-video-gen
npx clawhub@latest install veo3-video-gen
npx clawhub@latest install tube-cog
npx clawhub@latest install video-cog

# 语音合成服务 Skills
npx clawhub@latest install elevenlabs
npx clawhub@latest install azure-tts
```

```
npx clawhub@latest install google-tts
npx clawhub@latest install openai-tts
```text
**💡 重要提示**：
- ClawHub 市场的 Skills 在不断更新，某些 Skills 可能不存在或已更名
- 安装前建议先搜索：`npx clawhub@latest search <skill-name>`
- 如果 Skill 不存在，可以：
  1. 使用对应服务的官方 API 直接集成
  2. 自己创建自定义 Skill
  3. 在 GitHub 上搜索社区贡献的 Skills

### 分场景安装建议

**场景1：内容创作者（必装）**
```bash
npx clawhub@latest install fal-ai
npx clawhub@latest install elevenlabs
# Notion 建议使用 API 直接集成
```text
- fal-ai：快速生成配图
- elevenlabs：高质量配音
- Notion：使用官方 API 集成（见 10.2.2 节）

**场景2：视频UP主（推荐）**
```bash
npx clawhub@latest install video-agent
npx clawhub@latest install tube-cog
npx clawhub@latest install elevenlabs
```text
- video-agent：AI头像视频
- tube-cog：YouTube内容创作
- elevenlabs：专业配音

**场景3：知识工作者（推荐）**
```bash
npx clawhub@latest install openai-tts
# Notion 建议使用 API 直接集成
```text
- openai-tts：文档转音频
- Notion：使用官方 API 集成（见 10.2.2 节）

**场景4：营销人员（推荐）**
```bash
```

```
npx clawhub@latest install fal-ai
npx clawhub@latest install video-agent
npx clawhub@latest install elevenlabs
```text
- fal-ai：营销素材生成
- video-agent：产品介绍视频
- elevenlabs：广告配音

**场景5：教育工作者（推荐）**
```bash
npx clawhub@latest install fal-ai
npx clawhub@latest install video-cog
npx clawhub@latest install openai-tts
```text
- fal-ai：教学配图
- video-cog：课程视频制作
- openai-tts：有声课件

### 配置优先级

**必装Skills（优先级：⭐⭐⭐⭐⭐）**
- notion - 知识管理（免费）
- openai-tts - 语音合成（性价比高）
- fal-ai - AI绘画（功能全面）

**推荐Skills（优先级：⭐⭐⭐⭐）**
- video-agent - 视频生成（质量高）
- elevenlabs - 语音合成（最自然）
- notion-mcp - Notion高级功能

**可选Skills（优先级：⭐⭐⭐）**
- 根据具体需求选择安装
- 可以先试用再决定

### 成本预算参考

**月度成本估算**：

| 使用场景 | Skills组合 | API成本 | 总成本/月 |
|---------|-----------|---------|----------|
| 轻度使用 | 基础3个 | $5-10 | $5-10 |
| 中度使用 | 推荐6个 | $20-50 | $20-50 |
| 重度使用 | 全部安装 | $50-150 | $50-150 |
```
```

```
| 专业创作 | 高端组合 | $100-300 | $100-300 |

**各服务成本明细**：

| 服务类型 | 轻度使用 | 中度使用 | 重度使用 |
|---------|---------|---------|---------|
| AI绘图 | $2-5 | $10-20 | $30-50 |
| Notion | $0 | $0 | $0 |
| 视频生成 | $5-10 | $20-50 | $50-100 |
| 语音合成 | $2-5 | $10-20 | $20-50 |

**成本优化建议**：
```

✅ 使用国产模型（DeepSeek、Kimi）
✅ 批量处理任务
✅ 复用生成结果
✅ 设置Token限制
✅ 选择合适的质量等级
✅ 利用免费额度
✅ 批量购买享受折扣

常见问题

Q1：Skills 安装失败怎么办？
```bash
# 检查网络连接
ping github.com

# 使用国内镜像
npm config set registry https://registry.npmmirror.com

# 重试安装
npx clawhub@latest install <skill-name>

# 查看详细错误
npx clawhub@latest install <skill-name> --verbose
```text
**Q2：Skills加载失败怎么办？**
```bash
# 查看Skills列表
openclaw skills list

# 检查Skills状态
openclaw skills check

# 查看Skill详细信息
openclaw skills info <skill-name>

# 重启OpenClaw
openclaw restart
```text
**Q3：API调用失败怎么办？**
```bash
# 检查API配置
openclaw config get api

# 测试API连接
openclaw api test <service-name>

# 查看错误日志
openclaw logs --tail 50 --filter <service-name>

# 验证API 密钥
```

```
openclaw api verify <service-name>
```text
**Q4：如何更新Skills？**
```bash
# 更新单个Skill
npx clawhub@latest update <skill-name>

# 更新所有Skills
npx clawhub@latest update --all

# 查看可更新的Skills
npx clawhub@latest list --outdated

# 查看更新日志
npx clawhub@latest changelog <skill-name>
```text
**Q5：如何卸载不需要的Skills？**
```bash
# 卸载单个Skill
npx clawhub@latest uninstall <skill-name>

# 卸载多个Skills
npx clawhub@latest uninstall <skill1> <skill2> <skill3>

# 检查Skills依赖
openclaw skills check
```text
### 最佳实践

**1. 渐进式安装**
```

第1周：安装基础Skills（notion, openai-tts）
熟悉基本使用，建立工作流程

第2周：添加创作Skills（fal-ai, video-agent）
根据需求扩展功能

第3周：优化配置和成本
调整参数，提升效率

第4周：建立自动化工作流
整合所有Skills，实现自动化
```

2. 定期维护

```bash
# 每周检查更新
npx clawhub@latest list --outdated

# 定期检查Skills状态
openclaw skills check

# 每季度备份配置
openclaw backup create

# 每半年评估使用情况
openclaw stats --period 6m
```

```text
**3. 性能优化**
```

```bash
# 只加载需要的Skills
openclaw config set skills.lazy-load true

# 设置并发限制
openclaw config set skills.max-concurrent 3

# 启用缓存
openclaw config set skills.cache.enabled true
openclaw config set skills.cache.ttl 3600

# 设置超时时间
openclaw config set skills.timeout 30000
```

```text
**4. 成本监控**
```

```bash
# 查看API使用统计
openclaw stats api --period 1m

# 设置成本预警
openclaw config set cost.alert.enabled true
openclaw config set cost.alert.threshold 100

# 查看成本报告
openclaw cost report --period 1m

# 导出成本数据
```

```
openclaw cost export --format csv
```text
### 进阶技巧

**技巧1：使用配置文件管理Skills**
```bash
# 创建内容创作配置
cat > ~/.openclaw/profiles/content-creation.json <<EOF
{
  "skills": ["fal-ai", "video-agent", "elevenlabs", "notion"],
  "description": "内容创作工具集"
}
EOF

# 创建知识管理配置
cat > ~/.openclaw/profiles/knowledge-management.json <<EOF
{
  "skills": ["notion", "notion-mcp", "openai-tts"],
  "description": "知识管理工具集"
}
EOF
```

技巧2：自定义快捷命令

```
# 创建快捷命令
openclaw alias create "画图" "使用fal-ai生成图片"
openclaw alias create "记笔记" "保存到Notion"
openclaw alias create "配音" "使用elevenlabs转语音"
openclaw alias create "做视频" "使用video-agent生成视频"

# 使用快捷命令
你：画图 1个可爱的小龙虾
你：记笔记 今天学习了API集成
你：配音 这段文字
你：做视频 产品介绍脚本
```text
**技巧3：批量安装脚本**

由于 `clawhub` 一次只能安装1个 Skill，可以创建批量安装脚本：

```bash
# 创建批量安装脚本
cat > install_skills.sh << 'EOF'
#!/bin/bash

# 定义要安装的Skills
skills=(
    "fal-ai"
    "notion"
    "elevenlabs"
    "video-agent"
)

echo "开始批量安装 Skills..."

for skill in "${skills[@]}"; do
    echo "正在安装: $skill"
    npx clawhub@latest install "$skill"

    if [ $? -eq 0 ]; then
        echo "✅ $skill 安装成功"
    else
        echo "❌ $skill 安装失败"
    fi
    echo "---"
done
```

```
echo "批量安装完成！"
EOF

chmod +x install_skills.sh
./install_skills.sh
```text
```bash
# 批量生成图片
openclaw batch run "fal-ai" \
  --input "prompts.txt" \
  --output "images/" \
  --parallel 3

# 批量转换音频
openclaw batch run "elevenlabs" \
  --input "texts/" \
  --output "audios/" \
  --parallel 5

# 批量导入Notion
openclaw batch run "notion" \
  --input "notes/" \
  --database "db-id" \
  --parallel 2
```

技巧4：工作流自动化

```
# 创建自动化工作流
openclaw workflow create "daily-content" \
  --trigger "schedule:09:00" \
  --steps "
    1. 生成今日主题（ask）
    2. 生成配图（fal-ai）
    3. 生成视频（video-agent）
    4. 生成配音（elevenlabs）
    5. 保存到Notion（notion）
  "

# 启用工作流
openclaw workflow enable "daily-content"

# 查看工作流状态
openclaw workflow status "daily-content"
```text
### 故障排查

**问题1：Skills无法加载**

可能原因：
- 依赖未安装
- 配置文件错误
- 权限不足

解决方案：
```bash
# 查看Skill依赖
openclaw skills info <skill-name>

# 重新安装依赖
cd ~/.openclaw/skills/<skill-name>
npm install

# 检查配置
openclaw config validate

# 修复权限
chmod -R 755 ~/.openclaw/skills/
```text
**问题2：API调用超时**
```
```

```
可能原因：
- 网络不稳定
- API服务器繁忙
- 超时设置过短

解决方案：
```bash
# 增加超时时间
openclaw config set api.timeout 60000

# 使用代理
openclaw config set api.proxy "http://proxy:port"

# 重试机制
openclaw config set api.retry.enabled true
openclaw config set api.retry.max 3
```text
**问题3：成本超支**

可能原因：
- 使用频率过高
- 质量设置过高
- 未设置限制

解决方案：
```bash
# 设置每日限额
openclaw config set cost.daily-limit 10

# 降低质量设置
openclaw config set image.quality "standard"
openclaw config set video.quality "medium"

# 启用缓存
openclaw config set cache.enabled true

# 查看成本明细
openclaw cost detail --period 1d
```text
---

## 本章小结
```

```
本章详细介绍了OpenClaw与第三方API服务的集成方法，涵盖了4个核心应用场景：

### 核心内容回顾

**10.1 AI绘图服务集成** - 效率提升90%+
- 完整的bananapro-image-gen Skill安装和配置教程
- 从基础使用到批量生成的实战案例
- 提示词优化技巧和成本控制方法
- 推荐5个ClawHub市场的AI绘画Skills

**10.2 Notion数据同步** - 效率提升98%+
- 3个推荐的Notion集成Skills（notion, notion-mcp, notion-database）
- 完整的API配置和权限设置流程
- 自动创建笔记、批量导入、会议记录等实战案例
- 自定义模板和自动化工作流配置

**10.3 视频生成服务** - 效率提升90%+
- 5个推荐的视频生成Skills（video-agent, sora-video-gen等）
- HeyGen、Sora、Veo等主流服务的配置方法
- AI头像视频、批量生成、多语言版本等实战案例
- 自定义头像、视频后期处理等进阶技巧

**10.4 语音合成接入** - 效率提升95%+
- 4个推荐的语音合成Skills（elevenlabs, azure-tts等）
- ElevenLabs、Azure、Google等服务的对比和配置
- 文本转语音、有声读物、多语言配音等实战案例
- 自定义音色、情感控制、批量处理等进阶技巧

### 综合效率提升

**平均效率提升**：93%+
**平均成本节省**：95%+
**适用场景**：创作、管理、营销、教育

### 核心价值

**1. 能力扩展**
```

✅ AI绘图：从文字到图像
✅ 知识管理：自动化笔记和数据库
✅ 视频创作：从脚本到成片
✅ 语音合成：从文字到声音

```
**2. 效率提升**
```

✅ 绘图：30分钟 → 1分钟
✅ 笔记：5分钟 → 5秒
✅ 视频：2小时 → 10分钟
✅ 配音：1小时 → 2分钟

```
**3. 成本优化**
```

✅ 绘图成本：50 →0.1
✅ Notion：免费
✅ 视频成本：200 →2
✅ 配音成本：100 →0.3

```
**4. 质量保证**
```

✅ 专业级输出质量
✅ 多语言支持
✅ 批量处理能力
✅ 自动化工作流

实战技巧总结

1. Skills选择
- 根据需求选择合适的Skills
- 优先使用ClawHub市场的成熟Skills
- 关注成本和质量的平衡

2. 配置优化
- 正确配置API 密钥和权限
- 设置合理的默认参数
- 启用自动化规则

3. 成本控制
- 使用合适的质量等级
- 批量处理享受折扣
- 复用生成结果

4. 质量提升
- 优化输入内容（提示词、脚本、文本）
- 使用高质量模型
- 进行后期处理

下一步行动

1. **立即开始**：选择1个场景，安装对应的Skills
2. **实践验证**：通过实际案例测试效果
3. **优化流程**：根据使用情况调整配置
4. **扩展应用**：探索更多集成可能性

通过API集成，OpenClaw从1个AI助手升级为全能创作平台。掌握这些集成方法，你将拥有：
- 🎨 AI绘画能力
- 📝 自动化知识管理
- 🎬 视频创作能力
- 🎙️ 语音合成能力

开始使用这些强大的API集成功能，让OpenClaw成为你的超级创作助手！

下一章预告： 第11章 高级配置（多模型切换/成本优化/性能调优）

第11章 高级配置（多模型切换/成本优化/性能调优）

> 💡 **本章目标**：掌握OpenClaw的高级配置技巧，包括Antigravity Manager配置、多模型切换、成本优化和性能调优。

⚙️ 本章内容

- 11.1 Antigravity Manager完全配置指南
- 11.2 多模型切换策略
- 11.3 记忆搜索配置（Memory Search）
- 11.4 成本优化方案
- 11.5 性能调优技巧
- 11.6 模型提供商配置详解
- 11.7 工具系统详解
- 11.8 CLI 命令完整参考

11.1 Antigravity Manager完全配置指南

11.1.1 什么是Antigravity Manager?

定义：

Antigravity Manager是1个AI API代理工具，可以让你通过本地服务访问多个AI模型（Claude、Gemini、GPT等），统一管理API 密钥和请求。

项目地址：https://github.com/lbjlaq/Antigravity-Manager

为什么要用Antigravity Manager?

把OpenClaw和Antigravity Manager结合使用，你可以：

- ✅ **本地部署**：所有数据在本地处理，保护隐私
- ✅ **统一管理**：1个工具管理所有AI模型
- ✅ **成本控制**：使用自己的API 密钥，避免中间商加价
- ✅ **灵活切换**：随时切换不同的模型，无需修改代码
- ✅ **技能扩展**：通过ClawHub安装各种实用技能

11.1.2 系统要求和前置准备

系统要求：

- macOS 10.15+、Windows 10+、或Linux
- 至少4GB内存
- 稳定的网络连接

需要准备的东西：

1. Antigravity Manager安装包
2. AI模型的API Key（或独享账号）
3. 基本的命令行操作能力

11.1.3 安装Antigravity Manager

macOS用户

1. 访问Antigravity Manager Releases
2. 下载最新版本的 `.dmg` 文件
3. 双击 `.dmg` 文件，将应用拖入 `Applications` 文件夹
4. 打开应用（首次打开可能需要在「系统偏好设置 → 安全性与隐私」中允许）

Windows用户

1. 访问Antigravity Manager Releases
2. 下载最新版本的 `.exe` 安装包
3. 运行安装程序，按照提示完成安装

4. 启动Antigravity Manager

Linux用户

1. 访问[Antigravity Manager Releases]
2. 下载最新版本的`.AppImage`或`.deb`文件
3. 给予执行权限并运行：

```
chmod +x Antigravity-Manager-*.AppImage
./Antigravity-Manager-*.AppImage
```text
#### 验证安装

启动后，应用会在本地运行1个API服务，默认地址：`http://127.0.0.1:8045`

在浏览器中访问这个地址，如果能看到管理界面，说明安装成功。

### 11.1.4 配置AI模型账号

Antigravity Manager需要你提供AI模型的API 密钥才能工作。

#### 方案1：使用官方API

**Claude API**
1. 访问[Anthropic Console](https://console.anthropic.com/)
2. 注册账号并绑定信用卡
3. 创建API Key
4. 复制保存

**Gemini API**
1. 访问[Google AI Studio](https://makersuite.google.com/app/apikey)
2. 登录Google账号
3. 创建API Key
4. 复制保存

**OpenAI API**
1. 访问[OpenAI Platform](https://platform.openai.com/api-keys)
2. 注册账号并绑定信用卡
3. 创建API Key
4. 复制保存

#### 方案2：购买独享账号（推荐）

如果你不想自己申请API，可以购买独享账号：

**推荐**：学生账号Gemini 3 Pro独享账号12个月（支持反重力）

**优势：**
- ✅ 独享账号，无需担心限流
- ✅ 支持Antigravity Manager
```
```

- ✅ 12个月有效期
- ✅ 性价比高
- ✅ 即买即用

在Antigravity Manager中配置API Key

1. 打开Antigravity Manager管理界面
2. 点击「API Keys」
3. 选择对应的AI服务商（Claude、Gemini、OpenAI）
4. 输入API Key
5. 点击「保存」

11.1.5 生成User Token

User Token是OpenClaw访问Antigravity Manager的凭证。

1. 在Antigravity Manager界面中，点击右上角「User Tokens」
2. 点击「创建新Token」
3. 复制生成的Token（例如：`sk-82bc103b51f24af888af525a7835e87c`）
4. ⚠️ **重要**：妥善保存这个Token，它只会显示一次！

11.1.6 配置 OpenClaw

配置Claude Sonnet 4.5（默认模型）

这是最常用的模型，适合日常对话和代码生成。

```bash
# 添加local-anthropic provider
cat ~/.openclaw/openclaw.json | jq '.models.providers["local-anthropic"] = {
  "baseUrl": "http://127.0.0.1:8045",
  "apiKey": "你的User_Token",
  "auth": "api-key",
  "api": "anthropic-messages",
  "models": [
    {
      "id": "claude-sonnet-4-5-20250929",
      "name": "Local Claude Sonnet 4.5",
      "reasoning": false,
      "input": ["text"],
      "cost": {
        "input": 0,
        "output": 0,
```

```
        "cacheRead": 0,
        "cacheWrite": 0
      },
      "contextWindow": 200000,
      "maxTokens": 8192
    }
  ]
}' > /tmp/openclaw-temp.json && mv /tmp/openclaw-temp.json
~/.openclaw/openclaw.json

# 设置为默认模型
openclaw config set agents.defaults.model.primary "local-anthropic/claude-sonnet-
4-5-20250929"
```text
**注意**：把`你的User_Token`替换成第三步生成的Token。

#### 配置Claude Opus 4.5 Thinking（推理模型）

这是Claude的推理模型，适合复杂问题和深度思考。

```bash
cat ~/.openclaw/openclaw.json | jq '.models.providers["local-anthropic-opus"] = {
  "baseUrl": "http://127.0.0.1:8045",
  "apiKey": "你的User_Token",
  "auth": "api-key",
  "api": "anthropic-messages",
  "models": [
    {
      "id": "claude-opus-4-5-thinking",
      "name": "Local Claude Opus 4.5 Thinking",
      "reasoning": true,
      "input": ["text"],
      "cost": {
        "input": 0,
        "output": 0,
        "cacheRead": 0,
        "cacheWrite": 0
      },
      "contextWindow": 200000,
      "maxTokens": 8192
    }
  ]
}' > /tmp/openclaw-temp.json && mv /tmp/openclaw-temp.json
```

~/.openclaw/openclaw.json

```text
#### 配置Gemini 3 Pro Image（多模态模型）

这是Google的多模态模型，支持图片识别和分析。

```bash
cat ~/.openclaw/openclaw.json | jq '.models.providers["local-google"] = {
  "baseUrl": "http://127.0.0.1:8045/v1beta",
  "apiKey": "你的User_Token",
  "auth": "api-key",
  "api": "google-generative-ai",
  "models": [
    {
      "id": "gemini-3-pro-image",
      "name": "Local Gemini 3 Pro Image",
      "reasoning": false,
      "input": ["text", "image"],
      "cost": {
        "input": 0,
        "output": 0,
        "cacheRead": 0,
        "cacheWrite": 0
      },
      "contextWindow": 2000000,
      "maxTokens": 8192
    }
  ]
}' > /tmp/openclaw-temp.json && mv /tmp/openclaw-temp.json
~/.openclaw/openclaw.json
```text
### 11.1.7 验证配置

#### 检查模型列表

```bash
openclaw models list
```text
你应该看到：
```

Model Input Ctx Local Auth Tags
local-anthropic/claude-sonnet-4-5-20250929 text 195k yes yes default

local-anthropic-opus/claude-opus-4-5-thinking text 195k yes yes configured
local-google/gemini-3-pro-image text,image 1953k yes yes configured

重启Gateway

```bash
openclaw gateway restart
```text
#### 测试连接

```bash
openclaw message send "你好，介绍一下你自己"
```text
如果能正常返回回复，说明配置成功。

### 11.1.8 使用方法

#### 使用默认模型（Claude Sonnet 4.5）

直接发送消息即可：

```bash
openclaw message send "写1个Python脚本，打印Hello World"
```text
#### 切换到Opus Thinking模型

适合需要深度思考的复杂问题：

```bash
openclaw config set agents.defaults.model.primary "local-anthropic-opus/claude-opus-4-5-thinking"
openclaw gateway restart
```text
#### 切换到Gemini Image模型

适合需要图片识别的场景：

```bash
openclaw config set agents.defaults.model.primary "local-google/gemini-3-pro-image"
openclaw gateway restart
```text
#### 临时使用特定模型

不修改默认配置，临时使用某个模型：

```bash
# 使用Opus Thinking
openclaw agent --model "local-anthropic-opus/claude-opus-4-5-thinking" --message "解释量子计算的原理"

# 使用Gemini Image
openclaw agent --model "local-google/gemini-3-pro-image" --message "分析这张图片" --image ./photo.jpg
```text
### 11.1.9 模型选择指南

#### Claude Sonnet 4.5

**适用场景：**
- 日常对话
- 代码生成
- 文档编写
- 快速问答

**特点：**
- 速度快
- 成本低
- 质量高
- 上下文窗口：200k tokens

#### Claude Opus 4.5 Thinking

**适用场景：**
- 复杂推理
- 数学问题
- 算法优化
- 深度分析

**特点：**
- 推理能力强
- 思考过程可见
- 适合复杂问题
- 上下文窗口：200k tokens

#### Gemini 3 Pro Image

**适用场景：**
```
```

- 图片识别
- 多模态任务
- 文档分析
- 设计评审

特点：
- 支持图片输入
- 超大上下文窗口
- 识别准确
- 上下文窗口：2000k tokens

11.1.10 高级配置

配置模型别名

给模型起1个好记的名字：

```bash
openclaw config set agents.defaults.models."local-anthropic/claude-sonnet-4-5-20250929".alias "我的Claude"
```text
#### 添加多个API Key

如果你有多个Antigravity账号，可以配置多个provider：

```bash
cat ~/.openclaw/openclaw.json | jq '.models.providers["local-anthropic-2"] = {
  "baseUrl": "http://127.0.0.1:8045",
  "apiKey": "另1个User_Token",
  "auth": "api-key",
  "api": "anthropic-messages",
  "models": [...]
}' > /tmp/openclaw-temp.json && mv /tmp/openclaw-temp.json ~/.openclaw/openclaw.json
```text
#### 配置成本追踪

虽然本地API成本为0，但你可以设置虚拟成本来追踪使用量：

```json
{
  "cost": {
    "input": 0.003,
```

```
    "output": 0.015,
    "cacheRead": 0.0003,
    "cacheWrite": 0.00375
  }
}
```text
#### 备份配置

```bash
cp ~/.openclaw/openclaw.json ~/.openclaw/openclaw.json.backup
```text
#### 恢复配置

```bash
cp ~/.openclaw/openclaw.json.backup ~/.openclaw/openclaw.json
openclaw gateway restart
```text
### 11.1.11 常用命令速查

```bash
# 查看模型列表
openclaw models list

# 查看当前默认模型
openclaw config get agents.defaults.model.primary

# 切换默认模型
openclaw config set agents.defaults.model.primary "模型ID"

# 重启Gateway
openclaw gateway restart

# 查看配置文件
cat ~/.openclaw/openclaw.json | jq '.models.providers'

# 发送消息
openclaw message send "你的消息"

# 临时使用特定模型
openclaw agent --model "模型ID" --message "你的消息"
```text
```
```

```
### 11.1.12 模型ID速查
```

local-anthropic/claude-sonnet-4-5-20250929
local-anthropic-opus/claude-opus-4-5-thinking
local-google/gemini-3-pro-image

11.1.13 故障排查

问题1：模型列表为空

原因：配置文件格式错误或路径不对

解决方法：
```bash
# 检查配置文件
cat ~/.openclaw/openclaw.json | jq '.models.providers'

# 如果返回错误，恢复备份
cp ~/.openclaw/openclaw.json.backup ~/.openclaw/openclaw.json
```text
#### 问题2：API连接失败

**原因**：Antigravity Manager未启动或端口被占用

**解决方法**：
```bash
# 检查API是否正常
curl http://127.0.0.1:8045/v1/models

# 检查端口占用（macOS/Linux）
lsof -i :8045

# 重启Antigravity Manager
```text
#### 问题3：配置后模型不生效

**原因**：忘记重启Gateway

**解决方法**：
```bash
openclaw gateway restart
```text
#### 问题4：User Token无效

**原因**：Token过期或输入错误

**解决方法**：
1. 在Antigravity Manager中重新生成Token

```
2. 更新配置文件中的apiKey
3. 重启Gateway

---# 测试连接
openclaw test api
```

10.1.5 实战案例

案例1：配置Claude Sonnet

```
步骤：
1. 获取Claude API Key
2. 在Antigravity Manager中添加
3. 配置 OpenClaw
4. 测试使用

结果：
你：你好
OpenClaw（Claude Sonnet）：你好！我是Claude...
```

案例2：多账号管理

```
场景：管理多个Claude账号

配置：
- Claude账号1：日常使用
- Claude账号2：备用
- Claude账号3：高峰期使用

优势：
- 分散负载
- 避免限流
- 提高可用性
```

11.2 多模型切换策略

11.2.1 模型特点对比

模型	优势	劣势	适用场景
Claude Sonnet	平衡性好	价格中等	日常对话
Claude Opus	能力最强	价格最贵	复杂任务
GPT-5.2	功能丰富	响应较慢	创意工作
Gemini 3 Pro	免费额度大	能力一般	简单任务
DeepSeek-V3	性价比高	中文优化	编程任务

11.2.2 场景化选择策略

日常对话：

```
推荐：Claude Sonnet 4.5
理由：
- 响应速度快
- 质量稳定
- 价格适中
```text
**复杂推理**：
```text
推荐：Claude Opus 4.6
理由：
- 推理能力最强
- 准确率最高
- 适合难题
```text
**图片识别**：
```text
推荐：Gemini 3 Pro
理由：
- 多模态能力强
- 免费额度大
- 识别准确
```text
**编程任务**：
```text
推荐：DeepSeek-V3
理由：
- 代码能力强
- 价格便宜
- 中文友好
```text
### 11.2.3 模型容灾机制（Fallback）

> 🛡️ **高可用保障**：通过配置主模型和备用模型，确保服务不中断。

#### 什么是模型容灾？

当主模型（primary）出现以下情况时，系统会自动切换到备用模型（fallbacks）：
- API 调用失败
- 请求超时
- 速率限制（Rate Limit）
- 服务不可用
```
```

![服务容灾配置示例](https://upload.maynor1024.live/file/1771085328347_service-disaster-recovery.png)

基础容灾配置

配置文件路径：`~/.openclaw/openclaw.json`

```json
{
  "agents": {
    "defaults": {
      "model": {
        "primary": "anthropic/claude-opus-4-6",
        "fallbacks": [
          "openai-codex/gpt-5.3-codex",
          "google-antigravity/claude-opus-4-6-thinking"
        ]
      }
    },
    "list": [
      {
        "id": "main",
        "default": true,
        "model": {
          "primary": "anthropic/claude-opus-4-6",
          "fallbacks": [
            "openai-codex/gpt-5.3-codex",
            "google-antigravity/claude-opus-4-6-thinking"
          ]
        }
      }
    ]
  }
}
```text
**工作流程**：
```text
1. 尝试使用主模型：anthropic/claude-opus-4-6
   ↓ 失败
2. 切换到备用模型1：openai-codex/gpt-5.3-codex
   ↓ 失败
3. 切换到备用模型2：google-antigravity/claude-opus-4-6-thinking
   ↓ 失败
```

4. 返回错误信息
```text
#### 实战案例1：成本优化型容灾

**场景**：优先使用便宜模型，失败后使用高质量模型

```json
{
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat",
        "fallbacks": [
          "anthropic/claude-sonnet-4-5",
          "anthropic/claude-opus-4-6"
        ]
      }
    }
  }
}
```text
**优势**：
- ✅ 日常使用 DeepSeek（极低成本）
- ✅ DeepSeek 限流时自动切换到 Claude Sonnet
- ✅ 重要任务失败时使用 Claude Opus 兜底
- ✅ 成本节省 80%+

#### 实战案例2：性能优先型容灾

**场景**：优先使用最强模型，失败后降级

```json
{
  "agents": {
    "defaults": {
      "model": {
        "primary": "anthropic/claude-opus-4-6",
        "fallbacks": [
          "anthropic/claude-sonnet-4-5",
          "deepseek/deepseek-chat"
        ]
      }
    }
```
```

```
  }
}
```text
**优势**：
- ✅ 保证最佳质量
- ✅ 高峰期自动降级
- ✅ 确保服务不中断

#### 实战案例3：多提供商容灾

**场景**：跨提供商容灾，避免单点故障

```json
{
  "agents": {
    "defaults": {
      "model": {
        "primary": "anthropic/claude-sonnet-4-5",
        "fallbacks": [
          "openai/gpt-4o",
          "google/gemini-2.0-flash-exp",
          "deepseek/deepseek-chat"
        ]
      }
    }
  }
}
```text
**优势**：
- ✅ Anthropic 故障时切换到 OpenAI
- ✅ OpenAI 故障时切换到 Google
- ✅ 最后使用 DeepSeek 兜底
- ✅ 最大化服务可用性

#### 配置命令行方式

```bash
# 设置主模型
openclaw config set agents.defaults.model.primary "anthropic/claude-opus-4-6"

# 设置备用模型（需要手动编辑 JSON）
# 或使用 jq 命令
cat ~/.openclaw/openclaw.json | jq '.agents.defaults.model.fallbacks = [
```

```
  "openai-codex/gpt-5.3-codex",
  "google-antigravity/claude-opus-4-6-thinking"
]' > /tmp/openclaw-temp.json && mv /tmp/openclaw-temp.json
~/.openclaw/openclaw.json

# 重启 Gateway 使配置生效
openclaw gateway restart
```text
#### 验证容灾配置

```bash
# 查看当前配置
openclaw config get agents.defaults.model

# 输出示例：
{
  "primary": "anthropic/claude-opus-4-6",
  "fallbacks": [
    "openai-codex/gpt-5.3-codex",
    "google-antigravity/claude-opus-4-6-thinking"
  ]
}
```text
#### 容灾最佳实践

**1. 选择不同提供商**：
```text
✅ 推荐：Anthropic → OpenAI → Google
❌ 不推荐：Claude Opus → Claude Sonnet（同提供商）
```text
**2. 按能力梯度配置**：
```text
✅ 推荐：高能力 → 中能力 → 低能力
❌ 不推荐：低能力 → 高能力（浪费资源）
```text
**3. 考虑成本因素**：
```text
✅ 推荐：便宜 → 中等 → 昂贵（成本优化）
✅ 推荐：昂贵 → 中等 → 便宜（质量优先）
```text
**4. 限制备用数量**：
```text
✅ 推荐：2-3 个备用模型
```

❌ 不推荐：5+ 个备用模型（过度复杂）

```text
### 11.2.4 多认证 Profile + Token 轮换

> 🔐 **账号管理**：配置多个认证配置，实现账号轮换和负载均衡。

#### 什么是认证 Profile？

认证 Profile 允许你为同一个提供商配置多个账号，系统会按照指定顺序轮换使用，避免单账号限流。

#### 基础配置

**配置文件路径**：`~/.openclaw/openclaw.json`

```json
{
  "auth": {
    "profiles": {
      "openai-codex:default": {
        "provider": "openai-codex",
        "mode": "oauth"
      },
      "anthropic:default": {
        "provider": "anthropic",
        "mode": "token"
      },
      "anthropic:manual": {
        "provider": "anthropic",
        "mode": "token"
      },
      "google-antigravity:mail1@gmail.com": {
        "provider": "google-antigravity",
        "mode": "oauth",
        "email": "mail1@gmail.com"
      },
      "google-antigravity:mail2@gmail.com": {
        "provider": "google-antigravity",
        "mode": "oauth"
      }
    },
    "order": {
      "anthropic": [
        "anthropic:default",
```
```

```
        "anthropic:manual"
      ],
      "google-antigravity": [
        "google-antigravity:mail1@gmail.com",
        "google-antigravity:mail2@gmail.com"
      ]
    }
  }
}
```text
#### 配置说明

**profiles 字段**：
- 定义所有可用的认证配置
- 格式：`"提供商:标识符"`
- `mode`：认证方式（`oauth` 或 `token`）
- `email`：OAuth 账号邮箱（可选）

**order 字段**：
- 定义每个提供商的账号使用顺序
- 系统会按顺序轮换使用
- 当前账号限流时自动切换到下一个

#### 实战案例1：Anthropic 双账号轮换

**场景**：配置 2 个 Claude API Key，避免限流

```json
{
  "auth": {
    "profiles": {
      "anthropic:account1": {
        "provider": "anthropic",
        "mode": "token"
      },
      "anthropic:account2": {
        "provider": "anthropic",
        "mode": "token"
      }
    },
    "order": {
      "anthropic": [
        "anthropic:account1",
```

```
        "anthropic:account2"
      ]
    }
  }
}
```text
**配置 API Key**：
```bash
# 在 Antigravity Manager 中分别配置两个 API Key
# 或在 OpenClaw 配置中添加：
{
  "models": {
    "providers": {
      "anthropic": {
        "apiKey": "sk-ant-api-key-1",
        ...
      },
      "anthropic-2": {
        "apiKey": "sk-ant-api-key-2",
        ...
      }
    }
  }
}
```text
**工作流程**：
```text
1. 使用 account1 发送请求
2. account1 达到限流 → 自动切换到 account2
3. account2 达到限流 → 等待 account1 恢复
4. 循环往复
```text
#### 实战案例2：Google 多邮箱轮换

**场景**：使用多个 Google 账号访问 Gemini

```json
{
  "auth": {
    "profiles": {
      "google-antigravity:work@gmail.com": {
        "provider": "google-antigravity",
        "mode": "oauth",
```

```
        "email": "work@gmail.com"
      },
      "google-antigravity:personal@gmail.com": {
        "provider": "google-antigravity",
        "mode": "oauth",
        "email": "personal@gmail.com"
      },
      "google-antigravity:backup@gmail.com": {
        "provider": "google-antigravity",
        "mode": "oauth",
        "email": "backup@gmail.com"
      }
    },
    "order": {
      "google-antigravity": [
        "google-antigravity:work@gmail.com",
        "google-antigravity:personal@gmail.com",
        "google-antigravity:backup@gmail.com"
      ]
    }
  }
}
```text
**优势**：
- ✅ 3 个账号轮换，限流概率降低 66%
- ✅ 免费额度叠加（3 倍免费额度）
- ✅ 高峰期自动负载均衡

#### 实战案例3：混合认证模式

**场景**：同时使用 OAuth 和 API Token

```json
{
  "auth": {
    "profiles": {
      "anthropic:oauth-account": {
        "provider": "anthropic",
        "mode": "oauth"
      },
      "anthropic:token-account": {
        "provider": "anthropic",
        "mode": "token"
```

```
      }
    },
    "order": {
      "anthropic": [
        "anthropic:oauth-account",
        "anthropic:token-account"
      ]
    }
  }
}
```text
**使用场景**：
- OAuth 账号：日常使用（更安全）
- Token 账号：备用（更稳定）

#### 配置最佳实践

**1. 账号数量建议**：
```text
✅ 推荐：2-3 个账号
❌ 不推荐：5+ 个账号（管理复杂）
```text
**2. 认证方式选择**：
```text
OAuth：更安全，适合个人账号
Token：更稳定，适合 API 密钥
```text
**3. 轮换策略**：
```text
✅ 按使用频率排序（高频 → 低频）
✅ 按账号等级排序（付费 → 免费）
```text
**4. 监控和维护**：
```bash
# 查看当前使用的认证配置
openclaw config get auth.profiles

# 测试认证是否有效
openclaw test api
```text
### 11.2.5 自动切换配置

**基于任务类型切换**：
```
```

```json
{
  "rules": [
    {
      "condition": "task.type === 'code'",
      "model": "deepseek-v3"
    },
    {
      "condition": "task.type === 'image'",
      "model": "gemini-3-pro"
    },
    {
      "condition": "task.complexity === 'high'",
      "model": "claude-opus-4.6"
    },
    {
      "condition": "default",
      "model": "claude-sonnet-4.5"
    }
  ]
}
```text
**基于成本切换**：
```json
{
  "rules": [
    {
      "condition": "cost.daily < 10",
      "model": "claude-opus-4.6"
    },
    {
      "condition": "cost.daily >= 10",
      "model": "claude-sonnet-4.5"
    }
  ]
}
```text
---

## 11.3 记忆搜索配置（Memory Search）

> 🧠 **智能记忆**：配置 Memory Search 让 OpenClaw 记住历史对话，提供更智能的上下文感知。

### 11.3.1 什么是 Memory Search？

Memory Search 是 OpenClaw 的记忆系统，可以：
- 记住历史对话内容
- 搜索相关会话记录
- 提供上下文感知
- 支持混合检索（向量 + 文本）

### 11.3.2 基础配置

**配置文件路径**：`~/.openclaw/openclaw.json`

```json
{
  "agents": {
    "defaults": {
      "memorySearch": {
        "sources": ["memory", "sessions"],
        "experimental": {
          "sessionMemory": true
        },
        "provider": "gemini",
        "remote": {
          "apiKey": "AIzaSy************************"
        },
        "fallback": "gemini",
        "model": "gemini-embedding-001",
        "query": {
          "hybrid": {
            "enabled": true,
            "vectorWeight": 0.7,
            "textWeight": 0.3
          }
        }
      }
    }
  }
}
```text
### 11.3.3 配置项详解

#### sources（数据源）

```json
{
  "sources": ["memory", "sessions"]
}
```text
**可选值**：
- `memory`：长期记忆（跨会话）
- `sessions`：会话记录（当前会话）

**推荐配置**：
```json
// 只使用长期记忆
"sources": ["memory"]

// 同时使用长期记忆和会话记录
"sources": ["memory", "sessions"]
```text
#### experimental（实验性功能）

```json
{
  "experimental": {
    "sessionMemory": true
  }
}
```text
**sessionMemory**：
- `true`：启用会话记忆（推荐）
- `false`：禁用会话记忆

#### provider（嵌入模型提供商）

```json
{
  "provider": "gemini"
}
```text
**支持的提供商**：
- `gemini`：Google Gemini（推荐，免费）
- `openai`：OpenAI Embeddings
- `local`：本地嵌入模型

**推荐**：使用 Gemini（免费且效果好）

#### remote（远程 API 配置）

```json
{
  "remote": {
    "apiKey": "AIzaSy************************"
  }
}
```text
**获取 Gemini API Key**：
1. 访问 [Google AI Studio](https://makersuite.google.com/app/apikey)
2. 登录 Google 账号
3. 创建 API Key
4. 复制并粘贴到配置中

#### fallback（备用提供商）

```json
{
  "fallback": "gemini"
}
```text
当主提供商失败时，使用备用提供商。

#### model（嵌入模型）

```json
{
  "model": "gemini-embedding-001"
}
```text
**Gemini 嵌入模型**：
- `gemini-embedding-001`：标准模型（推荐）
- `text-embedding-004`：高级模型

**OpenAI 嵌入模型**：
- `text-embedding-3-small`：小模型（便宜）
- `text-embedding-3-large`：大模型（效果好）

#### query（查询配置）

```json
```

```
{
  "query": {
    "hybrid": {
      "enabled": true,
      "vectorWeight": 0.7,
      "textWeight": 0.3
    }
  }
}
```text
**hybrid（混合检索）**：
- `enabled`：是否启用混合检索
- `vectorWeight`：向量搜索权重（0-1）
- `textWeight`：文本搜索权重（0-1）

**权重建议**：
```text
语义搜索优先：vectorWeight: 0.7, textWeight: 0.3
关键词搜索优先：vectorWeight: 0.3, textWeight: 0.7
平衡模式：vectorWeight: 0.5, textWeight: 0.5
```text
### 11.3.4 实战案例1：基础配置（Gemini）

**场景**：使用免费的 Gemini 嵌入模型

```json
{
  "agents": {
    "defaults": {
      "memorySearch": {
        "sources": ["memory", "sessions"],
        "experimental": {
          "sessionMemory": true
        },
        "provider": "gemini",
        "remote": {
          "apiKey": "你的Gemini_API_Key"
        },
        "model": "gemini-embedding-001",
        "query": {
          "hybrid": {
            "enabled": true,
            "vectorWeight": 0.7,
```

```
            "textWeight": 0.3
          }
        }
      }
    }
  }
}
```text
**优势**：
- ✅ 完全免费
- ✅ 效果优秀
- ✅ 配置简单

### 11.3.5 实战案例2：高级配置（OpenAI）

**场景**：使用 OpenAI 嵌入模型（更高精度）

```json
{
  "agents": {
    "defaults": {
      "memorySearch": {
        "sources": ["memory", "sessions"],
        "experimental": {
          "sessionMemory": true
        },
        "provider": "openai",
        "remote": {
          "apiKey": "sk-your-openai-api-key"
        },
        "fallback": "gemini",
        "model": "text-embedding-3-large",
        "query": {
          "hybrid": {
            "enabled": true,
            "vectorWeight": 0.8,
            "textWeight": 0.2
          }
        }
      }
    }
  }
}
```

```text
**优势**：
- ✅ 精度更高
- ✅ 支持更多语言
- ✅ 有备用方案

**成本**：
- text-embedding-3-small：$0.02/百万 tokens
- text-embedding-3-large：$0.13/百万 tokens

### 11.3.6 实战案例3：本地部署（隐私优先）

**场景**：使用本地嵌入模型，保护隐私

```json
{
  "agents": {
    "defaults": {
      "memorySearch": {
        "sources": ["memory", "sessions"],
        "experimental": {
          "sessionMemory": true
        },
        "provider": "local",
        "model": "all-MiniLM-L6-v2",
        "query": {
          "hybrid": {
            "enabled": true,
            "vectorWeight": 0.6,
            "textWeight": 0.4
          }
        }
      }
    }
  }
}
```text
**优势**：
- ✅ 完全本地，保护隐私
- ✅ 无需 API Key
- ✅ 无使用限制

**劣势**：
```

- ❌ 需要本地计算资源
- ❌ 精度略低于云端模型

11.3.7 配置命令行方式

```bash
# 启用 Memory Search
openclaw config set agents.defaults.memorySearch.experimental.sessionMemory true

# 设置提供商
openclaw config set agents.defaults.memorySearch.provider "gemini"

# 设置 API Key（需要手动编辑 JSON）
# 或使用 jq 命令
cat ~/.openclaw/openclaw.json | jq '.agents.defaults.memorySearch.remote.apiKey =
"你的API_Key"' > /tmp/openclaw-temp.json && mv /tmp/openclaw-temp.json
~/.openclaw/openclaw.json

# 重启 Gateway
openclaw gateway restart
```text
### 11.3.8 验证配置

```bash
# 查看当前配置
openclaw config get agents.defaults.memorySearch

# 测试记忆搜索
openclaw message send "记住：我喜欢喝咖啡"
openclaw message send "我喜欢喝什么？"

# 应该返回：根据我的记忆，你喜欢喝咖啡。
```text
### 11.3.9 使用场景

**场景1：个人助手**
```text
你：记住我的生日是 1990 年 1 月 1 日
OpenClaw：好的，已记住。

（几天后）
你：我的生日是什么时候？
OpenClaw：根据我的记忆，你的生日是 1990 年 1 月 1 日。
```

```text
**场景2：项目管理**
```text
你：项目 A 的截止日期是 2026 年 3 月 1 日
OpenClaw：已记录。

（一周后）
你：项目 A 什么时候截止？
OpenClaw：项目 A 的截止日期是 2026 年 3 月 1 日。
```text
**场景3：知识积累**
```text
你：DeepSeek API 的价格是 $0.001/千 tokens
OpenClaw：已记住。

（下次对话）
你：哪个模型最便宜？
OpenClaw：根据我的记忆，DeepSeek 最便宜，价格是 $0.001/千 tokens。
```text
### 11.3.10 最佳实践

**1. 选择合适的提供商**：
```text
免费用户：Gemini（免费且效果好）
付费用户：OpenAI（精度更高）
隐私优先：Local（完全本地）
```text
**2. 调整混合检索权重**：
```text
语义理解为主：vectorWeight: 0.7-0.8
关键词匹配为主：textWeight: 0.6-0.7
平衡模式：各 0.5
```text
**3. 定期清理记忆**：
```bash
# 清理过期记忆
openclaw memory clean --older-than 30d

# 查看记忆使用情况
openclaw memory stats
```text
**4. 备份重要记忆**：
```bash
```
```

```
# 导出记忆
openclaw memory export --output memory-backup.json

# 导入记忆
openclaw memory import memory-backup.json
```text
### 11.3.11 故障排查

**问题1：记忆搜索不工作**

**原因**：API Key 无效或未配置

**解决方法**：
```bash
# 检查配置
openclaw config get agents.defaults.memorySearch

# 测试 API Key
curl -H "Content-Type: application/json" \
  -d '{"contents":[{"parts":[{"text":"test"}]}]}' \
  "https://generativelanguage.googleapis.com/v1beta/models/gemini-embedding-
001:embedContent?key=你的API_Key"
```text
**问题2：搜索结果不准确**

**原因**：混合检索权重不合适

**解决方法**：
```json
// 调整权重
{
  "query": {
    "hybrid": {
      "vectorWeight": 0.8,  // 提高语义搜索权重
      "textWeight": 0.2
    }
  }
}
```text
**问题3：记忆占用空间过大**

**原因**：长期积累未清理
```
```

```
**解决方法**：
```bash
# 查看记忆大小
openclaw memory stats

# 清理旧记忆
openclaw memory clean --older-than 60d

# 压缩记忆数据库
openclaw memory compact
```text
---

## 11.4 成本优化方案

### 11.4.1 Token消耗分析

**查看消耗统计**：
```bash
# 查看今日消耗
openclaw stats today

# 输出示例：
今日Token消耗：
- Claude Sonnet：150K tokens ($0.75)
- Gemini Pro：50K tokens ($0.00)
- 总计：200K tokens ($0.75)

任务分布：
- 文件搜索：30%
- 日程管理：20%
- 知识管理：25%
- 其他：25%
```text
**消耗优化建议**：
```

⚠️ 高消耗任务：

- 文件搜索：每次10K tokens
- 建议：优化搜索范围

✅ 优化方案：

- 使用缓存

- 减少上下文
- 优化提示词

```
### 11.4.2 缓存策略

**启用缓存**：
```bash
# 启用响应缓存
openclaw config set cache.enabled true

# 设置缓存时间（小时）
openclaw config set cache.ttl 24

# 设置缓存大小（MB）
openclaw config set cache.maxSize 1000
```text
**缓存效果**：
```

未启用缓存：

- 相同问题每次都调用API
- Token消耗：10K/次
- 成本：$0.05/次

启用缓存后：

- 相同问题直接返回缓存
- Token消耗：0
- 成本：$0
- 节省：100%

```
### 11.4.3 模型降级方案

**降级策略**：
```

1. 简单任务用便宜模型
2. 复杂任务用贵模型
3. 失败后降级重试

```
**配置示例**：
```javascript
{
  "fallback": [
    "claude-opus-4.6",    // 首选
    "claude-sonnet-4.5",  // 降级1
    "gemini-3-pro"        // 降级2
  ]
}
```text
### 11.4.4 成本控制实战

**案例1：降低50%成本**
```

原方案：

- 全部使用Claude Opus
- 日均消耗：$20

优化方案：

- 简单任务用Sonnet
- 复杂任务用Opus
- 启用缓存

优化后：

- 日均消耗：$10
- 节省：50%

```
**案例2：免费额度最大化**
```

策略：

1. 优先使用Gemini（免费额度大）
2. 超额后切换到DeepSeek（便宜）
3. 重要任务用Claude

效果：

- 月成本：$5
- 节省：90%

```
---

## 11.5 性能调优技巧

### 11.5.1 响应速度优化

**优化前**：
```

平均响应时间：5秒
用户体验：一般

```
**优化方案**：
```

1. 启用缓存
2. 减少上下文
3. 使用流式输出
4. 并发处理

```
**优化后**：
```

平均响应时间：2秒
用户体验：优秀
提升：60%

```
### 11.5.2 并发处理优化

**配置并发数**：
```bash
# 设置最大并发数
openclaw config set concurrency.max 5

# 设置队列大小
openclaw config set concurrency.queueSize 100
```text
### 11.5.3 内存管理

**监控内存使用**：
```bash
# 查看内存使用
openclaw stats memory

# 输出示例：
内存使用情况：
- 当前：512MB
- 峰值：800MB
- 平均：600MB
```text
**优化建议**：
```

⚠️ 内存占用高：

- 清理缓存
- 减少并发
- 重启服务

📝 本章小结

学习了OpenClaw的高级配置：
1. Antigravity Manager配置
2. 多模型切换策略
3. 成本优化方案
4. 性能调优技巧

掌握这些技巧可以：
- 降低50%以上成本
- 提升60%响应速度
- 提高系统稳定性

11.6 模型提供商配置详解

> 🤖 **多模型支持**：OpenClaw 支持 20+ 主流 AI 模型提供商，灵活配置满足不同需求。

11.6.0 快速配置：使用命令行向导（推荐新手）

> 💡 **最简单的方式**：使用 `openclaw onboard` 命令启动配置向导，交互式配置模型。

启动配置向导

```bash
openclaw onboard
```text
执行后会启动命令行交互式配置向导。

#### 配置流程

**步骤1：选择初始化模式**

```text
◇  初始化模式
│  快速开始
```text
**步骤2：选择模型提供商**

```text
◆  模型/认证提供商
|  ○ OpenAI (Codex OAuth + API key)
|  ○ Anthropic
|  ○ MiniMax
|  ○ Moonshot AI
|  ○ Google
|  ○ OpenRouter
|  ○ Qwen
|  ○ Z.AI (GLM 4.7)
|  ○ Copilot
|  ○ Vercel AI Gateway
|  ○ OpenCode Zen
|  ○ Xiaomi
|  ○ Synthetic
|  ○ Venice AI
|  ○ Skip for now
```text
使用 **方向键** 选择，**空格键** 确认。

**步骤3：输入 API Key**

根据提示输入对应提供商的 API Key。

**步骤4：选择默认模型**

从可用模型列表中选择默认模型。

**步骤5：完成配置**

配置自动保存并重启 Gateway。

#### 命令行向导的优势

✅ **交互式操作**：逐步引导，不易出错
✅ **实时验证**：输入 API Key 后立即验证有效性
✅ **自动配置**：自动生成配置文件
✅ **一键保存**：自动保存并重启服务
✅ **错误提示**：配置错误时会有明确的提示信息

#### 验证配置

配置完成后，验证模型是否可用：
```
```

```bash
# 查看已配置的模型
openclaw models list

# 测试模型连接
openclaw message send "你好，测试一下"
```text
#### 修改配置

如果需要修改配置，再次运行：

```bash
openclaw onboard
```text
可以添加、删除或修改模型提供商。

---

### 11.6.1 支持的模型提供商

#### 国际模型

| 提供商 | 模型 | 特点 | 价格 |
|--------|------|------|------|
| **OpenAI** | GPT-4o, GPT-4o-mini | 功能全面、生态完善 | 高 |
| **Anthropic** | Claude 3.5 Sonnet, Claude 3 Opus | 推理能力强、安全性高 | 中高 |
| **Google** | Gemini 2.0 Flash, Gemini 1.5 Pro | 多模态能力强、免费额度大 | 中 |
| **xAI** | Grok 2 | 实时信息、幽默风格 | 中 |
| **Mistral** | Mistral Large, Mistral Small | 开源友好、性价比高 | 中 |
| **Cohere** | Command R+, Command R | 企业级、RAG 优化 | 中 |

#### 国产模型

| 提供商 | 模型 | 特点 | 价格 |
|--------|------|------|------|
| **DeepSeek** | DeepSeek-V3, DeepSeek-Chat | 性价比之王、编程能力强 | 极低 |
| **月之暗面** | Kimi k2.5 | 超长上下文（200万字） | 低 |
| **智谱AI** | GLM-4, GLM-4V | 多模态、中文优化 | 中 |
| **百川智能** | Baichuan-4 | 中文理解好 | 中 |
| **MiniMax** | abab6.5 | 语音合成、角色扮演 | 中 |
| **阿里云** | Qwen-Max, Qwen-Plus | 阿里生态、企业级 | 中 |
| **百度** | ERNIE 4.0 | 百度生态、知识增强 | 中 |

#### 本地模型

| 提供商 | 模型 | 特点 | 价格 |
|--------|------|------|------|
| **Ollama** | Llama 3.1, Qwen2.5 | 完全本地、隐私保护 | 免费 |
| **LM Studio** | 各种开源模型 | 图形界面、易用 | 免费 |

### 11.6.2 配置 OpenAI

```json
{
  "models": {
    "mode": "merge",
    "providers": {
      "openai": {
        "baseUrl": "https://api.openai.com/v1",
        "apiKey": "sk-your-api-key",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "gpt-4o",
            "name": "GPT-4o",
            "contextWindow": 128000,
            "maxTokens": 16384
          },
          {
            "id": "gpt-4o-mini",
            "name": "GPT-4o Mini",
            "contextWindow": 128000,
            "maxTokens": 16384
          }
        ]
      }
    }
  }
}
```text
### 11.6.3 配置 Anthropic (Claude)

```json
{
```

```
  "models": {
    "mode": "merge",
    "providers": {
      "anthropic": {
        "baseUrl": "https://api.anthropic.com",
        "apiKey": "sk-ant-your-api-key",
        "auth": "api-key",
        "api": "anthropic",
        "models": [
          {
            "id": "claude-3-5-sonnet-20241022",
            "name": "Claude 3.5 Sonnet",
            "contextWindow": 200000,
            "maxTokens": 8192
          },
          {
            "id": "claude-3-opus-20240229",
            "name": "Claude 3 Opus",
            "contextWindow": 200000,
            "maxTokens": 4096
          }
        ]
      }
    }
  }
}
```text
### 11.6.4 配置 Google Gemini

```json
{
  "models": {
    "mode": "merge",
    "providers": {
      "google": {
        "baseUrl": "https://generativelanguage.googleapis.com/v1beta",
        "apiKey": "your-google-api-key",
        "auth": "api-key",
        "api": "google-ai",
        "models": [
          {
            "id": "gemini-2.0-flash-exp",
            "name": "Gemini 2.0 Flash",
```

```
            "contextWindow": 1000000,
            "maxTokens": 8192
          },
          {
            "id": "gemini-1.5-pro",
            "name": "Gemini 1.5 Pro",
            "contextWindow": 2000000,
            "maxTokens": 8192
          }
        ]
      }
    }
  }
}
```text
### 11.6.5 配置 DeepSeek（推荐）

```json
{
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-your-api-key",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "deepseek-chat",
            "name": "DeepSeek Chat",
            "contextWindow": 64000,
            "maxTokens": 4096
          },
          {
            "id": "deepseek-coder",
            "name": "DeepSeek Coder",
            "contextWindow": 64000,
            "maxTokens": 4096
          }
        ]
      }
    }
```

```
  }
}
```text
### 11.6.6 配置 Kimi（月之暗面）

```json
{
  "models": {
    "mode": "merge",
    "providers": {
      "moonshot": {
        "baseUrl": "https://api.moonshot.cn/v1",
        "apiKey": "sk-your-api-key",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "moonshot-v1-8k",
            "name": "Kimi k2.5 8K",
            "contextWindow": 8000,
            "maxTokens": 4096
          },
          {
            "id": "moonshot-v1-32k",
            "name": "Kimi k2.5 32K",
            "contextWindow": 32000,
            "maxTokens": 4096
          },
          {
            "id": "moonshot-v1-128k",
            "name": "Kimi k2.5 128K",
            "contextWindow": 128000,
            "maxTokens": 4096
          }
        ]
      }
    }
  }
}
```text
### 11.6.7 配置 Ollama（本地模型）

```json
```

```
{
  "models": {
    "mode": "merge",
    "providers": {
      "ollama": {
        "baseUrl": "http://localhost:11434",
        "auth": "none",
        "api": "ollama",
        "models": [
          {
            "id": "llama3.1:8b",
            "name": "Llama 3.1 8B",
            "contextWindow": 128000,
            "maxTokens": 4096
          },
          {
            "id": "qwen2.5:7b",
            "name": "Qwen 2.5 7B",
            "contextWindow": 32000,
            "maxTokens": 4096
          }
        ]
      }
    }
  }
}
```text
### 11.6.8 多提供商配置示例

```json
{
  "models": {
    "mode": "merge",
    "providers": {
      "deepseek": {
        "baseUrl": "https://api.deepseek.com",
        "apiKey": "sk-deepseek-key",
        "auth": "api-key",
        "api": "openai-chat",
        "models": [
          {
            "id": "deepseek-chat",
            "name": "DeepSeek Chat",
```

```
            "contextWindow": 64000,
            "maxTokens": 4096
          }
        ]
      },
      "anthropic": {
        "baseUrl": "https://api.anthropic.com",
        "apiKey": "sk-ant-key",
        "auth": "api-key",
        "api": "anthropic",
        "models": [
          {
            "id": "claude-3-5-sonnet-20241022",
            "name": "Claude 3.5 Sonnet",
            "contextWindow": 200000,
            "maxTokens": 8192
          }
        ]
      },
      "ollama": {
        "baseUrl": "http://localhost:11434",
        "auth": "none",
        "api": "ollama",
        "models": [
          {
            "id": "llama3.1:8b",
            "name": "Llama 3.1 8B",
            "contextWindow": 128000,
            "maxTokens": 4096
          }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "deepseek/deepseek-chat",
        "fallback": [
          "anthropic/claude-3-5-sonnet-20241022",
          "ollama/llama3.1:8b"
        ]
      }
```

```
    }
  }
}
```text
### 11.6.9 模型选择策略

**按任务类型选择**：

```javascript
// 编程任务
"deepseek/deepseek-coder"

// 长文档处理
"moonshot/moonshot-v1-128k"

// 复杂推理
"anthropic/claude-3-opus-20240229"

// 日常对话
"deepseek/deepseek-chat"

// 多模态（图片）
"google/gemini-2.0-flash-exp"

// 本地隐私
"ollama/llama3.1:8b"
```text
**按成本选择**：
```

极低成本：DeepSeek ($0.001/千tokens$)低成本：$Kimi, GLM-4$(0.01/千tokens)
中等成本：Gemini, Mistral ($0.05/千tokens$)高成本：$Claude, GPT-4$(0.15/千tokens)
免费：Ollama（本地）
```

```
---

## 11.7 工具系统详解

> 🔧 **扩展能力**：OpenClaw 的工具系统让 AI 能够执行各种操作，从文件管理到 API 调用。

### 11.7.1 内置工具列表

#### 文件系统工具

| 工具 | 功能 | 示例 |
|------|------|------|
| `read_file` | 读取文件内容 | 读取配置文件 |
| `write_file` | 写入文件 | 保存笔记 |
| `list_directory` | 列出目录 | 查看文件列表 |
| `search_files` | 搜索文件 | 找到所有 PDF |
| `move_file` | 移动文件 | 整理文件 |
| `delete_file` | 删除文件 | 清理临时文件 |

#### Shell 工具

| 工具 | 功能 | 示例 |
|------|------|------|
| `execute_command` | 执行命令 | 运行脚本 |
| `run_script` | 运行脚本 | 批处理任务 |

#### 网络工具

| 工具 | 功能 | 示例 |
|------|------|------|
| `web_search` | 网页搜索 | 搜索最新信息 |
| `fetch_url` | 获取网页 | 下载内容 |
| `api_call` | API 调用 | 调用第三方服务 |

#### 数据处理工具

| 工具 | 功能 | 示例 |
|------|------|------|
| `parse_json` | 解析 JSON | 处理 API 响应 |
| `parse_csv` | 解析 CSV | 处理表格数据 |
| `extract_text` | 提取文本 | 从 PDF 提取 |
```

11.7.2 启用和禁用工具

查看可用工具：
```bash
openclaw tools list
```text
**启用工具**：
```bash
openclaw tools enable read_file write_file
```text
**禁用工具**：
```bash
openclaw tools disable execute_command
```text
**配置文件方式**：
```json
{
  "tools": {
    "enabled": [
      "read_file",
      "write_file",
      "list_directory",
      "web_search"
    ],
    "disabled": [
      "execute_command",
      "delete_file"
    ]
  }
}
```text
### 11.7.3 工具权限控制

**设置工具权限**：
```json
{
  "tools": {
    "permissions": {
      "read_file": {
        "allowedPaths": [
          "~/Documents",
          "~/Downloads"
        ],
```

```
        "deniedPaths": [
          "~/.ssh",
          "~/.openclaw"
        ]
      },
      "execute_command": {
        "allowedCommands": [
          "ls",
          "cat",
          "grep"
        ],
        "deniedCommands": [
          "rm",
          "sudo"
        ]
      }
    }
  }
}
```text
### 11.7.4 自定义工具开发

**创建自定义工具**：

```javascript
// ~/.openclaw/tools/my-tool.js
export default {
  name: "my_custom_tool",
  description: "我的自定义工具",
  parameters: {
    type: "object",
    properties: {
      input: {
        type: "string",
        description: "输入参数"
      }
    },
    required: ["input"]
  },
  async execute({ input }) {
    // 工具逻辑
    return {
      success: true,
```

```
      result: `处理结果: ${input}`
    };
  }
};
```text
**注册工具**：
```bash
openclaw tools register ~/.openclaw/tools/my-tool.js
```text
### 11.7.5 工具使用示例

**文件搜索**：
```

你：帮我找到所有包含"发票"的 PDF 文件

OpenClaw 使用工具：

1. search_files(pattern="*.pdf", content="发票")
2. 返回结果：找到 3 个文件
   - 发票_2024_01.pdf
   - 报销发票.pdf
   - 采购发票_Q1.pdf

```
**网页搜索**：
```

你：Claude 3.5 Sonnet 最新价格是多少?

OpenClaw 使用工具：

1. web_search(query="Claude 3.5 Sonnet pricing")
2. fetch_url(url="https://www.anthropic.com/pricing")
3. 返回结果：
   - 输入：$3/百万 tokens
   - 输出：$15/百万 tokens

```
**数据处理**：
```

你：分析这个 CSV 文件的销售数据

OpenClaw 使用工具：

1. read_file(path="sales.csv")
2. parse_csv(content=...)
```

3. 分析数据并生成报告

```
### 11.7.6 工具链（Tool Chaining）

OpenClaw 可以自动组合多个工具完成复杂任务：
```

任务：下载网页并保存为 Markdown

工具链：

1. fetch_url(url) → 获取网页内容
2. extract_text(html) → 提取文本
3. convert_to_markdown(text) → 转换格式
4. write_file(path, content) → 保存文件

11.7.7 工具安全最佳实践

1. 最小权限原则：
```json
{
  "tools": {
    "enabled": [
      "read_file",  // 只启用必要的工具
      "web_search"
    ]
  }
}
```text
**2. 路径限制**：
```json
{
  "tools": {
    "permissions": {
      "read_file": {
        "allowedPaths": ["~/Documents"]  // 限制访问范围
      }
    }
  }
}
```text
**3. 命令白名单**：
```json
{
  "tools": {
    "permissions": {
      "execute_command": {
        "allowedCommands": ["ls", "cat"]  // 只允许安全命令
      }
    }
  }
}
```text
---

## 11.8 CLI 命令完整参考

> 📟 **命令行工具**：OpenClaw 提供强大的 CLI 工具，方便管理和操作。

### 11.8.1 核心命令

#### 版本和帮助

```bash
# 查看版本
openclaw --version
openclaw -v

# 查看帮助
openclaw --help
openclaw -h

# 查看子命令帮助
openclaw gateway --help
```text
#### 初始化和配置

```bash
# 运行配置向导
openclaw onboard

# 快速开始向导
openclaw setup

# 查看配置
openclaw config list

# 获取配置项
openclaw config get models.providers

# 设置配置项
openclaw config set gateway.port 18790

# 删除配置项
openclaw config delete models.providers.test
```text
### 11.8.2 Gateway 管理

```bash
# 安装/启动 Gateway
openclaw gateway install
```

```bash
# 查看状态
openclaw gateway status

# 停止 Gateway
openclaw gateway stop

# 重启 Gateway
openclaw gateway restart

# 查看日志
openclaw logs
openclaw logs --follow
openclaw logs --tail 100

# 清理日志
openclaw logs clear
```text
### 11.8.3 渠道管理

```bash
# 列出所有渠道
openclaw channels list

# 查看渠道状态
openclaw channels status

# 添加渠道
openclaw channels add

# 删除渠道
openclaw channels remove feishu

# 测试渠道
openclaw channels test feishu
```text
### 11.8.4 配对管理

```bash
# 列出配对请求
openclaw pairing list
openclaw pairing list feishu
```

```bash
# 批准配对
openclaw pairing approve feishu <CODE>

# 拒绝配对
openclaw pairing reject feishu <CODE>

# 清理过期配对
openclaw pairing cleanup
```text
### 11.8.5 插件管理

```bash
# 列出已安装插件
openclaw plugins list

# 搜索插件
openclaw plugins search feishu

# 安装插件
openclaw plugins install @openclaw/feishu

# 卸载插件
openclaw plugins uninstall @openclaw/feishu

# 更新插件
openclaw plugins update @openclaw/feishu

# 更新所有插件
openclaw plugins update --all
```text
### 11.8.6 工具管理

```bash
# 列出所有工具
openclaw tools list

# 启用工具
openclaw tools enable read_file write_file

# 禁用工具
openclaw tools disable execute_command

# 注册自定义工具
```

```
openclaw tools register ~/my-tool.js

# 测试工具
openclaw tools test read_file
```text
### 11.8.7 Agent 管理

```bash
# 列出 Agents
openclaw agents list

# 创建 Agent
openclaw agents create my-agent

# 删除 Agent
openclaw agents delete my-agent

# 切换 Agent
openclaw agents switch my-agent

# 查看 Agent 配置
openclaw agents config my-agent
```text
### 11.8.8 会话管理

```bash
# 列出会话
openclaw sessions list

# 查看会话详情
openclaw sessions show <session-id>

# 删除会话
openclaw sessions delete <session-id>

# 清理所有会话
openclaw sessions clear

# 导出会话
openclaw sessions export <session-id> --output session.json

# 导入会话
openclaw sessions import session.json
```

```text
### 11.8.9 统计和监控

```bash
# 查看统计信息
openclaw stats

# 查看今日统计
openclaw stats today

# 查看本周统计
openclaw stats week

# 查看 API 消耗
openclaw stats api

# 查看内存使用
openclaw stats memory

# 查看性能指标
openclaw stats performance
```text
### 11.8.10 测试和诊断

```bash
# 测试 API 连接
openclaw test api

# 测试渠道
openclaw test channel feishu

# 测试工具
openclaw test tool read_file

# 运行诊断
openclaw diagnose

# 检查配置
openclaw validate config

# 检查健康状态
openclaw health check
```text
```

11.8.11 数据管理

```bash
# 备份数据
openclaw backup create

# 列出备份
openclaw backup list

# 恢复备份
openclaw backup restore <backup-id>

# 清理缓存
openclaw cache clear

# 清理临时文件
openclaw cleanup temp

# 导出数据
openclaw export --output data.json

# 导入数据
openclaw import data.json
```text
### 11.8.12 更新和维护

```bash
# 检查更新
openclaw update check

# 更新到最新版本
openclaw update

# 更新到指定版本
openclaw update --version 2026.2.9

# 回滚版本
openclaw rollback

# 卸载
openclaw uninstall
```text
### 11.8.13 开发和调试

```bash
# 开发模式启动
openclaw dev

# 调试模式
openclaw --debug

# 详细日志
openclaw --verbose

# 运行测试
openclaw test

# 构建项目
openclaw build

# 清理构建
openclaw clean
```text
### 11.8.14 常用命令组合

**快速重启**：
```bash
openclaw gateway stop && openclaw gateway install
```text
**查看实时日志**：
```bash
openclaw logs --follow | grep ERROR
```text
**备份并更新**：
```bash
openclaw backup create && openclaw update
```text
**清理并重启**：
```bash
openclaw cache clear && openclaw gateway restart
```text
**完整诊断**：
```bash
openclaw diagnose && openclaw health check && openclaw test api
```text
### 11.8.15 环境变量
```
```

```bash
# 设置日志级别
export OPENCLAW_LOG_LEVEL=debug

# 设置配置目录
export OPENCLAW_HOME=~/.openclaw

# 设置 Gateway 端口
export OPENCLAW_PORT=18789

# 设置 API Key
export DEEPSEEK_API_KEY=sk-xxx
export MOONSHOT_API_KEY=sk-xxx
```text
### 11.8.16 配置文件位置

```bash
# 主配置文件
~/.openclaw/openclaw.json

# 日志文件
~/.openclaw/logs/gateway.log

# 缓存目录
~/.openclaw/cache/

# 数据目录
~/.openclaw/data/

# 插件目录
~/.openclaw/plugins/

# 工具目录
~/.openclaw/tools/
```

📝 本章小结

学习了OpenClaw的高级配置：

核心内容

1. **Antigravity Manager配置** - API 统一管理
2. **多模型切换策略** - 场景化选择 + 模型容灾机制
3. **记忆搜索配置** - 智能上下文感知
4. **成本优化方案** - 降低 50%+ 成本
5. **性能调优技巧** - 提升 60% 响应速度
6. **模型提供商配置** - 20+ 主流模型支持
7. **工具系统详解** - 扩展 AI 能力
8. **CLI 命令完整参考** - 100+ 命令详解

实战技能

- ✅ 配置多个 AI 模型提供商
- ✅ 配置模型容灾机制（primary + fallbacks）
- ✅ 配置多认证 Profile 实现账号轮换
- ✅ 配置记忆搜索系统
- ✅ 根据任务选择最优模型
- ✅ 使用工具系统扩展功能
- ✅ 掌握 CLI 命令高效管理
- ✅ 优化成本和性能

推荐配置

- **日常使用**：DeepSeek（性价比最高）
- **长文档**：Kimi（200万字上下文）
- **复杂任务**：Claude 3.5 Sonnet（推理能力强）
- **本地隐私**：Ollama（完全本地）
- **容灾方案**：DeepSeek → Claude Sonnet → Claude Opus
- **记忆搜索**：Gemini Embedding（免费且效果好）

下一章预告：第12章将进入实战案例部分，学习个人效率提升的完整工作流。

第12章 5类人群的效率提升实战

> 💡 **本章目标**：通过5个真实场景，学习如何用OpenClaw打造个人效率提升系统，涵盖知识工作者、程序员、内容创作者、学生和超级个体的完整工作流。

🎯 本章内容

- 12.1 知识工作者高效工作流落地
- 12.2 程序员的开发助手实战
- 12.3 内容创作者的工作流优化
- 12.4 学生的学习助手应用
- 12.5 云上OpenClaw的5种正确打开方式

11.1 知识工作者的一天

11.1.1 场景描述

角色：张明，管理咨询顾问
工作特点：

- 需要处理大量信息
- 频繁的会议和沟通
- 需要快速产出报告
- 知识管理要求高

痛点：

```
❌ 信息分散：邮件、微信、文档到处都是
❌ 时间碎片：会议太多，难以专注
❌ 重复劳动：每天都在做相似的事
❌ 知识流失：看过的东西记不住
```

目标：

```
✅ 信息集中管理
✅ 自动化重复任务
✅ 提升专注时间
✅ 构建个人知识库
```

11.1.2 早晨：日报推送和日程提醒

7:00 AM - 起床后的第一件事

打开手机，OpenClaw已经推送了今日日报：

```
📊 今日日报（2026年2月10日）

【重要日程】
• 09:00-10:00 客户A项目启动会（线上）
• 14:00-16:00 行业研究报告讨论（会议室B）
• 17:00-18:00 团队周会（线上）

【待办事项】
• 完成客户B的需求分析报告
• 审核团队提交的市场调研数据
• 准备明天的客户提案PPT

【行业动态】
• AI行业：OpenAI发布新模型GPT-5.3
• 咨询行业：麦肯锡发布2026年趋势报告
• 竞品动态：竞争对手C公司获得新融资

【今日建议】
• 上午会议较多，建议提前准备材料
• 下午有2小时专注时间，可用于报告撰写
• 晚上建议复盘今日工作，更新知识库
```

配置方法：

```
# 1. 配置定时任务
openclaw schedule add "daily-report" \
  --time "07:00" \
  --prompt "生成今日日报，包括：日程、待办、行业动态、建议"

# 2. 配置信息源
openclaw config set report.sources \
  "calendar,todo,rss,notion"

# 3. 配置推送渠道
openclaw config set report.channel "feishu"
```text
**效果**：
- 每天早上7点自动推送
- 信息来源：日历、待办、RSS订阅、Notion笔记
- 推送到飞书，手机随时查看
- 节省时间：30分钟/天

![日报推送示例](https://upload.maynor1024.live/file/1770176303922_image_25.jpg)

### 11.1.3 上午：资料收集和整理

**9:00 AM - 客户项目启动会前**

需要快速了解客户所在行业的最新动态。

**传统方式**：
```

1. 打开浏览器搜索
2. 阅读多篇文章
3. 手动复制粘贴到文档
4. 整理成结构化笔记
   ⏱️ 耗时：1小时

```
**OpenClaw方式**：
```

你：帮我收集一下新能源汽车行业最近一周的重要新闻，
整理成结构化笔记，存到备忘录

OpenClaw：好的，正在收集...
```

【新能源汽车行业周报】
📅 2026年2月3日-2月10日

【政策动态】
- 工信部发布新能源汽车补贴新政
- 北京市放宽新能源车牌指标
- 欧盟通过2030年碳排放新标准

【企业动态】
- 特斯拉Q4财报超预期，股价上涨15%
- 比亚迪发布新款电池技术，续航提升30%
- 蔚来宣布进军欧洲市场

【技术趋势】
- 固态电池技术取得突破
- 自动驾驶L4级量产在即
- 充电桩建设加速，覆盖率提升

【市场数据】
- 1月新能源车销量同比增长45%
- 渗透率达到35%，创历史新高
- 二手车市场开始活跃

已保存到备忘录：新能源汽车行业周报_20260210
⏱️ 耗时：2分钟

配置方法：

```bash
# 1. 配置网页抓取Skills
clawhub install web-clipper

# 2. 配置备忘录同步
clawhub install notes-sync

# 3. 设置默认存储位置
openclaw config set notes.default "备忘录/行业研究"
```text
**效果对比**：
| 项目 | 传统方式 | OpenClaw方式 | 提升 |
|------|----------|--------------|------|
| 时间 | 60分钟 | 2分钟 | 96.7% |
| 质量 | 中等 | 高 | +30% |
| 结构化 | 需手动 | 自动 | 100% |
| 可搜索 | 困难 | 容易 | +80% |

![网页文章存档](https://upload.maynor1024.live/file/1770176259365_image_15.jpg)


### 11.1.4 中午：会议记录和总结

**12:00 PM - 上午会议结束后**

上午开了3个会，需要整理会议纪要。

**传统方式**：
```

1. 回忆会议内容
2. 翻看会议笔记
3. 整理成文档
4. 发送给相关人员
   ⏱️ 耗时：45分钟

```
**OpenClaw方式**：
```
```

你：帮我整理一下今天上午客户A项目启动会的会议纪要，
包括：会议时间、参会人员、讨论要点、行动项、下次会议时间

OpenClaw：好的，正在整理...

【会议纪要】客户A项目启动会

📅 会议时间：2026年2月10日 09:00-10:00
👥 参会人员：张明（主持）、客户A方3人、团队成员2人

📋 讨论要点：

1. 项目背景和目标
   - 客户希望在Q2完成数字化转型
   - 预算范围：500-800万
   - 核心诉求：提升运营效率30%

2. 项目范围
   - 业务流程优化
   - 系统选型和实施
   - 人员培训

3. 时间计划
   - 需求调研：2周
   - 方案设计：3周
   - 系统实施：8周
   - 上线培训：2周

✅ 行动项：
- 张明：本周五前完成需求调研计划（负责人：张明，截止：2月14日）
- 团队：下周一提交初步方案（负责人：李华，截止：2月17日）
- 客户：提供现有系统资料（负责人：客户A，截止：2月12日）

📅 下次会议：2月17日 14:00，讨论初步方案

已保存到：会议纪要/客户A项目启动会_20260210.md
已同步到：Notion工作区
⏱️ 耗时：3分钟

```
**配置方法**：

```bash
# 1. 配置会议记录模板
openclaw template add "meeting-notes" \
  --format "时间、参会人员、讨论要点、行动项、下次会议"

# 2. 配置自动同步
openclaw config set sync.targets "notion,feishu"

# 3. 配置提醒
openclaw config set reminder.action-items true
```text
**效果**：
- 会议结束后3分钟完成纪要
- 自动提取行动项并设置提醒
- 同步到Notion和飞书
- 节省时间：42分钟/会议

### 11.1.5 下午：报告撰写和审核

**14:00 PM - 专注时间**

需要完成客户B的需求分析报告。

**OpenClaw辅助写作流程**：
```

步骤1：收集资料
你：帮我从Notion中找出所有关于客户B的资料，
包括：会议纪要、调研数据、行业分析

OpenClaw：已找到15份相关资料，正在整理...

步骤2：生成大纲
你：根据这些资料，生成一份需求分析报告的大纲

OpenClaw：
【需求分析报告大纲】

一、项目背景
1.1 客户基本情况
1.2 行业现状分析
1.3 项目发起原因

二、现状分析
2.1 业务流程现状
2.2 系统架构现状
2.3 存在的问题

三、需求分析
3.1 业务需求
3.2 功能需求
3.3 非功能需求

四、解决方案建议
4.1 业务优化建议
4.2 系统改造建议
4.3 实施路径建议

五、预期效果
5.1 效率提升预期
5.2 成本节约预期
5.3 风险评估

步骤3：逐章撰写
你：帮我写第一章：项目背景

OpenClaw：正在撰写...

一、项目背景

1.1 客户基本情况
客户B是一家成立于2015年的制造企业...
（详细内容）

1.2 行业现状分析
当前制造行业正面临数字化转型的关键时期...
（详细内容）

1.3 项目发起原因
基于以下三个核心驱动因素...
（详细内容）

步骤4：审核优化
你：帮我检查一下这份报告，看看有没有逻辑问题或者遗漏

OpenClaw：已完成审核，发现以下问题：

1. 第二章缺少数据支撑，建议补充具体数字
2. 第三章需求优先级不够清晰，建议增加优先级排序
3. 第四章解决方案过于笼统，建议细化实施步骤
4. 全文缺少图表，建议增加流程图和数据图表

已生成优化建议文档

```
**效果对比**：
| 项目 | 传统方式 | OpenClaw辅助 | 提升 |
|------|----------|--------------|------|
| 资料收集 | 30分钟 | 2分钟 | 93.3% |
| 大纲设计 | 20分钟 | 3分钟 | 85% |
| 内容撰写 | 120分钟 | 40分钟 | 66.7% |
| 审核优化 | 30分钟 | 5分钟 | 83.3% |
| **总计** | **200分钟** | **50分钟** | **75%** |


### 11.1.6 晚上：知识归档和复盘

**19:00 PM - 下班后**

一天的工作结束，需要整理今天的收获。

**知识归档流程**：
```

你：帮我整理一下今天的工作，生成一份日报

OpenClaw：
【工作日报】2026年2月10日

📊 今日完成：
✅ 客户A项目启动会（已完成会议纪要）
✅ 客户B需求分析报告（已完成初稿）
✅ 团队周会（已同步行动项）
✅ 行业研究（已归档3篇文章）

📈 数据统计：
- 会议：3场，共4小时
- 专注工作：4小时
- 文档产出：2份，共8000字
- 知识归档：5条笔记

💡 今日收获：
- 学习了新能源汽车行业的最新趋势
- 掌握了客户B的核心需求
- 优化了会议纪要的模板

📝 待办事项：
- 明天需要完成客户B报告的终稿

- 准备客户提案PPT
- 跟进客户A的资料收集

🎯 明日计划：
- 09:00-12:00 完成客户B报告终稿
- 14:00-17:00 制作客户提案PPT
- 17:00-18:00 团队内部讨论

已保存到：工作日报/20260210.md
已同步到：Notion个人工作区

自动归档配置：

```bash
# 1. 配置自动归档规则
openclaw config set archive.rules '{
  "会议纪要": "Notion/会议记录",
  "行业研究": "备忘录/行业研究",
  "项目文档": "Notion/项目管理",
  "个人笔记": "备忘录/个人成长"
}'

# 2. 配置日报生成
openclaw schedule add "daily-summary" \
  --time "19:00" \
  --prompt "生成今日工作日报"

# 3. 配置知识图谱
openclaw config set knowledge.graph true
```text
### 11.1.7 完整工作流配置

**一键配置脚本**：

```bash
#!/bin/bash
# 知识工作者完整工作流配置

# 1. 早晨日报
openclaw schedule add "morning-report" \
  --time "07:00" \
  --prompt "生成今日日报" \
  --channel "feishu"

# 2. 网页剪藏
clawhub install web-clipper
openclaw config set clipper.default "备忘录/行业研究"

# 3. 会议记录
openclaw template add "meeting-notes"
openclaw config set meeting.auto-sync true

# 4. 文档协作
```

```
clawhub install notion-sync
openclaw config set notion.workspace "个人工作区"

# 5. 晚间复盘
openclaw schedule add "evening-summary" \
  --time "19:00" \
  --prompt "生成工作日报"

# 6. 知识图谱
openclaw config set knowledge.graph true
openclaw config set knowledge.auto-link true

echo "✅ 知识工作者工作流配置完成！"
```text
### 11.1.8 效率提升数据分析

**使用前后对比**：

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 信息收集 | 60分钟 | 5分钟 | 55分钟 | 91.7% |
| 会议纪要 | 45分钟 | 3分钟 | 42分钟 | 93.3% |
| 报告撰写 | 200分钟 | 50分钟 | 150分钟 | 75% |
| 知识归档 | 30分钟 | 5分钟 | 25分钟 | 83.3% |
| **每日总计** | **335分钟** | **63分钟** | **272分钟** | **81.2%** |

**每月效率提升**：
```

节省时间：272分钟/天 × 22天 = 5,984分钟 ≈ 100小时
相当于：12.5个工作日
年度节省：150个工作日（约7.5个月）

```
**投资回报率（ROI）**：
```

成本：

- OpenClaw 配置时间：4小时
- API费用：$50/月
- 总成本：$50/月

收益：

- 节省时间：100小时/月
```

- 按时薪50计算：5,000/月
- ROI：(5,000−50) / $50 = 9,900%

```
---

## 11.2 程序员的开发助手

### 11.2.1 场景描述

**角色**：李华，全栈开发工程师
**工作特点**：
- 需要管理大量代码
- 频繁查阅技术文档
- 需要追踪Bug和任务
- 持续学习新技术

**痛点**：
```

❌ 代码分散：多个项目，难以管理
❌ 文档查找：技术文档到处都是
❌ Bug追踪：问题记录不系统
❌ 学习碎片：学习资料难以整理

```
**目标**：
```

✅ 统一代码管理
✅ 快速查找文档
✅ 系统化Bug追踪
✅ 构建技术知识库

```
### 11.2.2 代码搜索和管理

**场景：快速定位代码**
```

你：帮我找一下项目中所有处理用户认证的代码

OpenClaw：正在搜索...

找到以下相关文件：

1. src/auth/login.js - 登录逻辑
2. src/auth/register.js - 注册逻辑

3. src/middleware/auth.js - 认证中间件
4. src/utils/jwt.js - JWT工具函数
5. tests/auth.test.js - 认证测试

核心代码片段：

```
// src/middleware/auth.js
const authMiddleware = async (req, res, next) => {
  const token = req.headers.authorization;
  if (!token) {
    return res.status(401).json({ error: 'No token provided' });
  }
  // ... 验证逻辑
}
```text
已生成代码地图：auth-code-map.md
```

配置方法：

```
# 1. 配置代码搜索
clawhub install code-search
openclaw config set code.paths "~/projects"

# 2. 配置代码分析
openclaw config set code.analysis true
openclaw config set code.language "javascript,python,go"

# 3. 配置代码地图
openclaw config set code.map.enabled true
```text
### 11.2.3 技术文档整理

**场景：学习新技术**

今天需要学习React 19的新特性。
```

你：帮我收集React 19的官方文档和社区最佳实践，
整理成学习笔记

OpenClaw：正在收集...

【React 19 学习笔记】

📚 官方文档：
- React 19 发布说明
- 新特性：Server Components
- 新特性：Actions
- 新特性：Document Metadata
- 迁移指南

🔥 核心新特性：

1. Server Components
   - 服务端渲染组件
   - 减少客户端JavaScript
   - 提升首屏加载速度
2. Actions
   - 简化表单处理
   - 自动处理pending状态
   - 内置错误处理
3. Document Metadata
   - 组件内直接设置meta标签
   - 更好的SEO支持

💡 最佳实践：
- 何时使用Server Components
- Actions的使用场景
- 性能优化建议

🔗 参考资源：
- 官方文档：https://react.dev/blog/2024/12/05/react-19
- 社区教程：10篇精选文章
- 示例代码：5个开源项目

已保存到：技术笔记/React19.md
已同步到：Notion技术知识库

```
**GitHub项目分析**：
```

你：帮我分析一下这个GitHub项目：
https://github.com/vercel/next.js

OpenClaw：正在分析...

【Next.js 项目分析】

📊 项目概况：
- Stars: 125K
- Forks: 26.8K
- 语言：TypeScript (95%)
- 最近更新：2小时前
- 活跃度：非常活跃

🏗️ 项目结构：

```
next.js/
├── packages/          # 核心包
│   ├── next/         # Next.js核心
│   ├── create-next-app/  # 脚手架
│   └── eslint-config-next/  # ESLint配置
├── examples/         # 示例项目（300+）
├── test/            # 测试用例
└── docs/            # 文档
```

🔑 核心功能：
- 服务端渲染（SSR）
- 静态站点生成（SSG）
- API路由
- 文件系统路由
- 图片优化
- 国际化支持

💻 技术栈：
- React 18+
- TypeScript
- Webpack/Turbopack
- SWC编译器

📈 最近更新：
- 新增：Turbopack支持
- 优化：构建速度提升40%
- 修复：15个Bug

🎯 适用场景：
- 企业级Web应用
- 电商网站
- 博客和内容网站
- SEO要求高的项目

已保存到：项目分析/NextJS.md

```
![GitHub项目分析](https://upload.maynor1024.live/file/1770176261237_image_16.jpg)

### 11.2.4 Bug追踪和解决

**场景：系统化管理Bug**
```

你：帮我记录一个Bug：
用户登录后，刷新页面会退出登录

OpenClaw：已记录Bug，正在分析...

【Bug报告】#2024-001

🐛 问题描述：
用户登录后，刷新页面会退出登录

📋 基本信息：
- 严重程度：高
- 影响范围：所有用户
- 发现时间：2026-02-10 15:30
- 报告人：李华

🔍 可能原因：

1. Token未正确存储到localStorage
2. 认证中间件逻辑错误
3. Session过期时间设置问题

💡 建议排查：

1. 检查login.js中的token存储逻辑
2. 检查auth.js中间件的token验证
3. 检查浏览器控制台的错误信息

📝 相关代码：
- src/auth/login.js:45-60
- src/middleware/auth.js:12-25

🔗 相关Issue：
- GitHub Issue #123（类似问题）
- Stack Overflow讨论

已创建任务：修复登录刷新Bug
已设置提醒：今天下班前处理
已保存到：Bug追踪/2024-001.md

```
**Bug解决流程**：
```

步骤1：分析代码
你：帮我分析一下login.js的token存储逻辑

OpenClaw：
发现问题：
第52行使用了sessionStorage而不是localStorage
sessionStorage在页面刷新后会清空

建议修改：

- sessionStorage.setItem('token', token);
- localStorage.setItem('token', token);

步骤2：修复验证
你：帮我检查修改后是否还有其他问题

OpenClaw：
已检查相关代码，发现：

1. ✅ Token存储已修复
2. ⚠️ 建议添加token过期检查
3. ⚠️ 建议添加错误处理

步骤3：更新文档
你：帮我更新Bug报告，标记为已解决

OpenClaw：
已更新Bug报告：
- 状态：已解决
- 解决方案：修改sessionStorage为localStorage
- 解决时间：2026-02-10 16:15
- 耗时：45分钟

已同步到：Bug追踪系统
已通知：相关团队成员

```
### 11.2.5 开发环境管理

**场景：快速切换项目环境**
```

你：帮我切换到项目A的开发环境

OpenClaw：正在切换...

✅ 已切换到项目A开发环境

📁 项目信息：
- 项目名：电商后台管理系统
- 路径：~/projects/ecommerce-admin
- 分支：feature/user-management
- Node版本：18.17.0
- 依赖状态：已安装

🔧 环境配置：
- 数据库：MySQL 8.0（本地）
- Redis：6.2（本地）
- API地址：http://localhost:3000

📝 最近修改：
- 3个文件已修改
- 2个文件待提交
- 0个冲突

💡 待办任务：
- 完成用户管理模块
- 修复登录Bug
- 编写单元测试

是否需要启动开发服务器？

```
**配置方法**：

```bash
# 1. 配置项目列表
openclaw config set projects '{
  "project-a": {
    "path": "~/projects/ecommerce-admin",
    "node": "18.17.0",
    "services": ["mysql", "redis"]
  },
  "project-b": {
    "path": "~/projects/blog-system",
    "node": "20.10.0",
    "services": ["postgresql"]
  }
}'

# 2. 配置自动启动
openclaw config set project.auto-start true

# 3. 配置环境检查
openclaw config set project.health-check true
```text
### 11.2.6 学习资源管理

**场景：系统化学习新技术**
```

你：帮我制定一个学习React 19的计划

OpenClaw：
【React 19 学习计划】

📅 学习周期：2周（2026-02-10 至 2026-02-24）

📚 第一周：基础学习
Day 1-2：Server Components
- 阅读官方文档
- 完成3个示例项目
- 总结学习笔记

Day 3-4：Actions
- 理解Actions概念

- 实践表单处理
- 对比传统方案

Day 5：Document Metadata
- 学习新API
- 实践SEO优化
- 周末复习总结

📚 第二周：实战应用
Day 6-8：重构现有项目
- 迁移到React 19
- 应用新特性
- 性能对比测试

Day 9-10：最佳实践
- 阅读社区文章
- 参与技术讨论
- 总结经验教训

🎯 学习目标：
- 掌握React 19核心特性
- 能够在项目中应用
- 输出技术分享文章

📊 进度追踪：
- 每日学习时间：2小时
- 每周复盘：周日晚上
- 最终考核：完成一个实战项目

已创建学习任务
已设置每日提醒
已同步到日历

11.2.7 完整工作流配置

程序员开发助手配置脚本：

```bash
#!/bin/bash
# 程序员开发助手完整配置

# 1. 代码搜索
clawhub install code-search
openclaw config set code.paths "~/projects"
openclaw config set code.language "javascript,typescript,python,go"

# 2. 技术文档管理
clawhub install web-clipper
clawhub install github-analyzer
openclaw config set docs.default "Notion/技术知识库"

# 3. Bug追踪
openclaw template add "bug-report"
openclaw config set bug.auto-create-task true
openclaw config set bug.priority-alert true

# 4. 项目环境管理
openclaw config set projects.list "project-a,project-b,project-c"
openclaw config set project.auto-start true

# 5. 学习计划
clawhub install learning-planner
openclaw config set learning.daily-reminder true

# 6. 代码审查
clawhub install code-review
openclaw config set review.auto-check true

echo "✅ 程序员开发助手配置完成！"
```text
### 11.2.8 开发效率提升数据

**使用前后对比**：

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |

```
|---------|--------|--------|----------|----------|
| 代码搜索 | 15分钟 | 2分钟 | 13分钟 | 86.7% |
| 文档查找 | 30分钟 | 3分钟 | 27分钟 | 90% |
| Bug追踪 | 20分钟 | 5分钟 | 15分钟 | 75% |
| 环境切换 | 10分钟 | 1分钟 | 9分钟 | 90% |
| 学习整理 | 60分钟 | 10分钟 | 50分钟 | 83.3% |
| **每日总计** | **135分钟** | **21分钟** | **114分钟** | **84.4%** |

**月度效率提升**：
```

节省时间：114分钟/天 × 22天 = 2,508分钟 ≈ 42小时
相当于：5.25个工作日
年度节省：63个工作日（约3个月）

```
---

## 11.3 内容创作者的工作流

### 11.3.1 场景描述

**角色**：王芳，自媒体创作者
**工作特点**：
- 需要持续产出内容
- 多平台运营
- 需要素材管理
- 数据分析优化

**痛点**：
```

❌ 选题困难：不知道写什么
❌ 素材分散：图片、文字到处都是
❌ 多平台发布：重复劳动
❌ 数据分析：缺乏系统化

```
**目标**：
```

✅ 系统化选题
✅ 统一素材管理
✅ 自动化发布
✅ 数据驱动优化

```
### 11.3.2 选题灵感收集

**场景：每日选题灵感**
```

你：帮我收集今天AI领域的热点话题，
生成5个选题建议

OpenClaw：正在收集...

【今日选题建议】2026-02-10

🔥 热点话题：

1. OpenAI发布GPT-5.3
   - 热度：⭐⭐⭐⭐⭐
   - 竞争度：高
   - 建议角度：深度评测、使用技巧
2. Claude 5泄露消息
   - 热度：⭐⭐⭐⭐
   - 竞争度：中
   - 建议角度：信息整理、预测分析
3. AI绘画新工具Banana Pro
   - 热度：⭐⭐⭐
   - 竞争度：低
   - 建议角度：使用教程、案例分享
4. 国产大模型价格战
   - 热度：⭐⭐⭐⭐
   - 竞争度：中
   - 建议角度：成本对比、选择指南
5. AI Agent应用案例
   - 热度：⭐⭐⭐
   - 竞争度：低
   - 建议角度：实战教程、效率提升

💡 推荐选题：
第3个（Banana Pro）
理由：
- 竞争度低，容易出圈
- 实用性强，用户需求大
- 可以做系列内容

📊 数据支持：
- 微信指数：上升50%
- 知乎热度：新增200+讨论
- B站搜索：增长80%

已保存到：选题库/20260210.md

```
**配置方法**：

```bash
# 1. 配置热点监控
clawhub install trend-monitor
openclaw config set trend.sources "weibo,zhihu,xiaohongshu"
openclaw config set trend.keywords "AI,人工智能,ChatGPT"

# 2. 配置选题生成
openclaw schedule add "daily-topics" \
  --time "09:00" \
  --prompt "生成今日选题建议"

# 3. 配置选题评分
openclaw config set topic.scoring true
openclaw config set topic.factors "热度,竞争度,实用性"
```text
### 11.3.3 资料搜集和整理

**场景：快速收集素材**
```

你：帮我收集Banana Pro的相关资料，
包括：官方介绍、使用教程、用户评价

OpenClaw：正在收集...

【Banana Pro 资料包】

📚 官方资料：
- 官网：https://banana.dev
- 产品介绍：AI绘画工具，支持文生图、图生图
- 核心特点：速度快、质量高、价格低
- 定价：$0.01/张

📖 使用教程：
- 官方文档：10篇教程

- 社区教程：15篇精选
- 视频教程：5个B站视频

💬 用户评价：
- 好评率：92%
- 主要优点：速度快、效果好、价格便宜
- 主要缺点：功能相对简单

🎨 示例作品：
- 已收集50张示例图片
- 分类：人物、风景、产品、插画

📊 竞品对比：
- vs Midjourney：价格优势明显
- vs Stable Diffusion：使用更简单
- vs DALL-E：速度更快

已整理成文档：资料库/BananaPro.md
已下载示例图片到：素材库/BananaPro/

```
![素材收集](https://upload.maynor1024.live/file/1770176324389_image_29.jpg)

### 11.3.4 内容创作辅助

**场景：撰写文章**
```

你：帮我写一篇关于Banana Pro的使用教程，
目标平台：公众号，字数：2000字

OpenClaw：正在创作...

【文章大纲】

一、Banana Pro是什么？
- 产品介绍
- 核心优势
- 适用人群

二、如何开始使用？
- 注册流程
- 基础设置
- 第一张图片

三、进阶技巧
- 提示词优化

- 参数调整
- 风格控制

四、实战案例
- 案例1：制作公众号封面
- 案例2：生成产品图
- 案例3：创作插画

五、常见问题
- Q&A汇总
- 避坑指南

【正文】（2000字）
一、Banana Pro是什么？

如果你是内容创作者，一定遇到过这样的困扰...
（详细内容）

【配图建议】
- 封面图：Banana Pro界面截图
- 图1：注册流程示意图
- 图2-4：实战案例效果图
- 图5：价格对比表

【SEO优化】
- 标题：Banana Pro使用教程：5分钟学会AI绘画
- 关键词：Banana Pro、AI绘画、使用教程
- 描述：详细的Banana Pro使用教程...

已生成文章：草稿/BananaPro教程.md
已生成配图需求：5张

```
**图片生成**：
```

你：帮我用Banana生成文章需要的5张配图

OpenClaw：正在生成...

✅ 已生成5张配图：

1. 封面图：Banana Pro界面（1200x630）
2. 注册流程图：步骤示意图（800x600）
3. 案例1：公众号封面效果（900x500）
4. 案例2：产品图效果（800x800）
5. 案例3：插画效果（1000x1000）

已保存到：素材库/BananaPro/配图/
已添加水印
已压缩优化

```
![AI绘画生成](https://upload.maynor1024.live/file/1770176332471_image_30.jpg)


### 11.3.5 多平台发布

**场景：一键多平台发布**
```

你：帮我把这篇文章发布到：
公众号、知乎、小红书、B站

OpenClaw：正在发布...

【发布进度】

✅ 公众号
- 标题：Banana Pro使用教程：5分钟学会AI绘画
- 封面：已上传
- 正文：已格式化
- 状态：已发布
- 链接：https://mp.weixin.qq.com/xxx

✅ 知乎
- 标题：同上
- 封面：已上传
- 正文：已适配知乎格式
- 话题：#AI绘画 #工具推荐
- 状态：已发布
- 链接：https://zhuanlan.zhihu.com/xxx

✅ 小红书
- 标题：5分钟学会AI绘画！Banana Pro保姆级教程
- 封面：已生成小红书风格封面
- 正文：已精简为1000字
- 话题：#AI绘画 #效率工具
- 状态：已发布
- 链接：https://xiaohongshu.com/xxx

⏳ B站（视频制作中）
- 脚本：已生成视频脚本

- 素材：已准备图片和音频
- 预计完成时间：2小时后

📊 发布统计：
- 已发布：3个平台
- 待发布：1个平台
- 总耗时：15分钟

```
**配置方法**：

```bash
# 1. 配置平台账号
openclaw config set platforms '{
  "wechat": {
    "appid": "xxx",
    "secret": "xxx"
  },
  "zhihu": {
    "token": "xxx"
  },
  "xiaohongshu": {
    "cookie": "xxx"
  }
}'

# 2. 配置自动适配
openclaw config set publish.auto-format true
openclaw config set publish.platforms "wechat,zhihu,xiaohongshu"

# 3. 配置发布模板
openclaw template add "multi-platform-publish"
```text
### 11.3.6 数据分析和优化

**场景：内容数据分析**
```

你：帮我分析一下上周发布的5篇文章的数据

OpenClaw：正在分析...

【内容数据分析】2026-02-03 至 2026-02-09

📊 整体数据：
- 发布文章：5篇
- 总阅读量：15,000
- 总点赞数：800
- 总收藏数：450
- 平均互动率：8.3%

📈 最佳表现：
文章：《Banana Pro使用教程》
- 阅读量：5,000（占比33.3%）
- 点赞数：350（占比43.8%）
- 收藏数：200（占比44.4%）
- 互动率：11%

📉 待优化：
文章：《AI行业周报》
- 阅读量：1,500（占比10%）
- 点赞数：50（占比6.3%）
- 互动率：3.3%

💡 优化建议：

1. 教程类内容表现最好，建议增加比例
2. 周报类内容互动率低，建议优化标题和封面
3. 发布时间：晚上8-9点效果最好
4. 标题：带数字和问号的标题表现更好

📊 平台对比：
- 公众号：阅读量最高，但互动率一般
- 知乎：互动率最高，但阅读量较低
- 小红书：增长最快，潜力最大

🎯 下周策略：
- 增加教程类内容（3篇）
- 优化周报形式（改为精选）
- 重点运营小红书
- 优化发布时间为晚上8点

已保存到：数据分析/20260210.md

11.3.7 完整工作流配置

内容创作者工作流配置脚本：

```bash
#!/bin/bash
# 内容创作者完整工作流配置

# 1. 选题灵感
clawhub install trend-monitor
openclaw schedule add "daily-topics" \
  --time "09:00" \
  --prompt "生成今日选题建议"

# 2. 资料收集
clawhub install web-clipper
clawhub install image-downloader
openclaw config set materials.path "~/素材库"

# 3. 内容创作
clawhub install content-writer
clawhub install banana-ai
openclaw config set content.style "公众号"

# 4. 多平台发布
clawhub install multi-publisher
openclaw config set publish.platforms "wechat,zhihu,xiaohongshu"

# 5. 数据分析
clawhub install analytics
openclaw schedule add "weekly-report" \
  --time "Mon 10:00" \
  --prompt "生成上周数据分析报告"

# 6. 素材管理
openclaw config set materials.auto-organize true
openclaw config set materials.backup true

echo "✅ 内容创作者工作流配置完成！"
```text
### 11.3.8 创作效率提升数据

```
**使用前后对比**：

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 选题策划 | 60分钟 | 10分钟 | 50分钟 | 83.3% |
| 资料收集 | 90分钟 | 15分钟 | 75分钟 | 83.3% |
| 内容创作 | 180分钟 | 60分钟 | 120分钟 | 66.7% |
| 图片制作 | 60分钟 | 10分钟 | 50分钟 | 83.3% |
| 多平台发布 | 45分钟 | 15分钟 | 30分钟 | 66.7% |
| 数据分析 | 30分钟 | 5分钟 | 25分钟 | 83.3% |
| **每篇总计** | **465分钟** | **115分钟** | **350分钟** | **75.3%** |

**月度效率提升**：
```

每周发布：3篇文章
节省时间：350分钟/篇 × 3篇 × 4周 = 4,200分钟 ≈ 70小时
相当于：8.75个工作日
年度节省：105个工作日（约5个月）

```
**收益提升**：
```

效率提升后：
- 发布频率：从每周3篇提升到每周5篇
- 内容质量：提升30%
- 粉丝增长：提升50%
- 广告收入：提升80%

```
---

## 11.4 学生的学习助手

### 11.4.1 场景描述

**角色**：陈晨，研究生二年级
**学习特点**：
- 需要阅读大量论文
- 课程作业繁重
- 科研项目管理
- 考试复习压力大

**痛点**：
```

❌ 论文太多：读不完、记不住
❌ 作业繁重：时间不够用
❌ 项目混乱：进度难以把控
❌ 复习低效：知识点太分散

```
**目标**：
```

✅ 高效阅读论文
✅ 快速完成作业
✅ 系统管理项目
✅ 科学复习备考

```
### 11.4.2 课程资料管理

**场景：整理课程资料**
```

你：帮我整理一下机器学习课程的所有资料，
包括：课件、作业、论文、笔记

OpenClaw：正在整理...

【机器学习课程资料库】

📚 课件（12份）
- 第1章：机器学习概述
- 第2章：线性回归
- 第3章：逻辑回归
- 第4章：神经网络
- ...

📝 作业（8份）
- 作业1：线性回归实现（已完成）
- 作业2：逻辑回归实现（已完成）
- 作业3：神经网络实现（进行中）
- 作业4：CNN实现（未开始）
- ...

📄 论文（25篇）
- 必读论文（10篇）

- AlexNet（已读）
- ResNet（已读）
- Transformer（未读）

- ...
  - 选读论文（15篇）

📖 笔记（30条）
- 按章节分类
- 按知识点分类
- 重点标记

已创建目录结构：
课程资料/
├── 课件/
├── 作业/
├── 论文/
│ ├── 必读/
│ └── 选读/
└── 笔记/

已生成学习进度表

```
### 11.4.3 论文阅读和笔记

**场景：快速阅读论文**
```

你：帮我读一下这篇论文，生成阅读笔记
论文：Attention Is All You Need（Transformer）

OpenClaw：正在阅读...

【论文笔记】Attention Is All You Need

📄 基本信息：
- 标题：Attention Is All You Need
- 作者：Vaswani et al.
- 发表：NIPS 2017
- 引用：50,000+
- 重要性：⭐⭐⭐⭐⭐

🎯 核心贡献：

1. 提出Transformer架构
2. 完全基于注意力机制
3. 抛弃了RNN和CNN
4. 并行化训练，速度更快

🔑 关键概念：
- Self-Attention：自注意力机制
- Multi-Head Attention：多头注意力
- Position Encoding：位置编码
- Feed-Forward Network：前馈网络

📊 模型架构：

```
Transformer
├── Encoder（6层）
│   ├── Multi-Head Attention
│   └── Feed-Forward Network
└── Decoder（6层）
    ├── Masked Multi-Head Attention
    ├── Multi-Head Attention
    └── Feed-Forward Network
```

💡 核心创新：

1. 注意力机制
   $Q = XW_Q, K = XW_K, V = XW_V$
   $\text{Attention}(Q,K,V) = \text{softmax}(QK^T/\sqrt{d_k})V$

2. 多头注意力
   允许模型关注不同位置的不同表示子空间

3. 位置编码
   $PE(pos,2i) = \sin(pos/10000^{(2i/d)})$
   $PE(pos,2i+1) = \cos(pos/10000^{(2i/d)})$

📈 实验结果：
- 机器翻译：BLEU提升2.0
- 训练速度：提升3.5倍
- 参数量：减少30%

🤔 个人思考：
- Transformer为什么有效？
- 如何应用到其他领域？
- 有哪些改进空间？

🔗 相关论文：
- BERT（2018）
- GPT（2018）
- Vision Transformer（2020）

已保存到：论文笔记/Transformer.md
已添加到：知识图谱
已创建：复习卡片

```
**配置方法**：

```bash
# 1. 配置论文阅读
clawhub install paper-reader
openclaw config set paper.language "中英文"
openclaw config set paper.detail-level "详细"

# 2. 配置笔记模板
openclaw template add "paper-notes" \
  --fields "基本信息,核心贡献,关键概念,个人思考"

# 3. 配置知识图谱
openclaw config set knowledge.graph true
openclaw config set knowledge.auto-link true
```text
### 11.4.4 作业辅助完成

**场景：编程作业辅助**
```

你：帮我完成机器学习作业3：
实现一个简单的神经网络，用于手写数字识别

OpenClaw：好的，让我帮你分析一下...

【作业分析】

📋 作业要求：
- 数据集：MNIST
- 模型：全连接神经网络
- 框架：PyTorch
- 要求：准确率>95%

💡 实现思路：

1. 数据加载和预处理
2. 定义网络结构
3. 训练模型
4. 评估性能

5. 可视化结果

📝 代码框架：

```
import torch
import torch.nn as nn
from torchvision import datasets, transforms

# 1. 数据加载
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

train_dataset = datasets.MNIST(
    './data', train=True, download=True, transform=transform
)

# 2. 定义网络
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(784, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 10)

    def forward(self, x):
        x = x.view(-1, 784)
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# 3. 训练代码
# （详细实现）

# 4. 评估代码
# （详细实现）
```text
🎯 关键点：
• 数据归一化很重要
• 学习率建议0.001
• Batch size建议64
• Epoch建议10

📊 预期结果：
```
```

- 训练准确率：98%
- 测试准确率：96%
- 训练时间：5分钟

⚠️ 注意事项：
- 记得保存模型
- 添加可视化代码
- 写好注释
- 完成实验报告

已生成完整代码：作业/作业3_神经网络.py
已生成实验报告模板：作业/作业3_报告.md

作业检查:

```
你：帮我检查一下代码有没有问题

OpenClaw：正在检查...

✅ 代码检查结果：

【语法检查】
✅ 无语法错误

【逻辑检查】
✅ 数据加载正确
✅ 网络结构合理
✅ 训练流程完整
⚠️ 建议添加验证集

【性能优化】
💡 建议使用GPU加速
💡 建议添加学习率衰减
💡 建议使用数据增强

【代码规范】
✅ 命名规范
✅ 注释完整
⚠️ 建议添加类型提示

【实验报告】
✅ 结构完整
✅ 图表清晰
⚠️ 建议补充实验分析

总体评分：92/100
预计成绩：A-

已生成优化建议：作业/作业3_优化建议.md
```

11.4.5 考试复习规划

场景：期末考试复习

你：帮我制定机器学习课程的期末复习计划，
考试时间：2周后

OpenClaw：正在制定...

【期末复习计划】机器学习

📅 复习周期：2周（2026-02-10 至 2026-02-24）
⏰ 每日学习：4小时
🎯 目标成绩：90+

📚 第一周：基础知识复习

Day 1-2：线性模型
- 线性回归
- 逻辑回归
- 正则化
- 复习时间：8小时
- 练习题：20道

Day 3-4：神经网络
- 前向传播
- 反向传播
- 激活函数
- 优化算法
- 复习时间：8小时
- 练习题：25道

Day 5-6：深度学习
- CNN
- RNN
- Transformer
- 复习时间：8小时
- 练习题：30道

Day 7：第一周总结
- 整理笔记
- 做综合练习
- 查漏补缺

📚 第二周：强化提升

```
Day 8-9：重点难点
• 梯度下降原理
• 过拟合与正则化
• 注意力机制
• 复习时间：8小时
• 练习题：20道

Day 10-11：真题练习
• 历年真题（3套）
• 模拟考试
• 错题整理
• 复习时间：8小时

Day 12-13：冲刺复习
• 知识点串讲
• 公式记忆
• 易错点总结
• 复习时间：8小时

Day 14：考前准备
• 快速过一遍笔记
• 放松心态
• 准备考试用品

📊 复习进度追踪：
• 每日打卡
• 每周测试
• 实时调整

⏰ 每日提醒：
• 09:00 开始复习
• 12:00 午休
• 14:00 继续复习
• 18:00 晚餐
• 19:00 练习题
• 21:00 总结复盘

已创建复习任务
已设置每日提醒
已同步到日历
已生成复习资料包
```

知识点卡片：

```
你：帮我生成所有重要知识点的复习卡片

OpenClaw：正在生成...

已生成120张复习卡片

【示例卡片1】
正面：什么是梯度下降？
背面：
梯度下降是一种优化算法，通过迭代更新参数来最小化损失函数。
公式：θ = θ - α∇J(θ)
其中：α是学习率，∇J(θ)是梯度

【示例卡片2】
正面：过拟合的解决方法有哪些？
背面：
1. 增加训练数据
2. 正则化（L1/L2）
3. Dropout
4. Early Stopping
5. 数据增强

【示例卡片3】
正面：Transformer的核心创新是什么？
背面：
1. Self-Attention机制
2. Multi-Head Attention
3. Position Encoding
4. 完全并行化

已保存到：复习卡片/机器学习.anki
可导入Anki使用
```

11.4.6 科研项目管理

场景：管理研究项目

你：帮我管理我的毕业论文项目，
包括：进度追踪、文献管理、实验记录

OpenClaw：好的，正在创建项目管理系统...

【毕业论文项目】

📋 项目信息：
- 题目：基于Transformer的文本分类研究
- 导师：张教授
- 截止时间：2026年6月1日
- 剩余时间：112天

📊 项目进度：
总体进度：35%

阶段1：文献调研（已完成）✅
- 阅读论文：50篇
- 文献综述：已完成
- 完成时间：2025年12月

阶段2：方案设计（进行中）⏳
- 模型设计：80%完成
- 实验方案：60%完成
- 预计完成：2026年2月底

阶段3：实验实施（未开始）
- 数据准备：0%
- 模型训练：0%
- 结果分析：0%
- 预计开始：2026年3月

阶段4：论文撰写（未开始）
- 初稿：0%
- 修改：0%
- 定稿：0%
- 预计开始：2026年5月

📚 文献管理：
- 已阅读：50篇
- 待阅读：15篇
- 重点文献：10篇

- 引用文献：30篇

🔬 实验记录：
- 实验1：基线模型（已完成）
  - 准确率：85%
  - 问题：过拟合

- 实验2：改进模型（进行中）
  - 当前准确率：88%
  - 优化方向：正则化

⚠️ 风险提示：
- 实验进度略慢，需要加快
- 数据集准备需要提前开始
- 建议每周与导师沟通

📅 本周任务：
- 完成模型设计文档
- 准备实验数据集
- 阅读3篇相关论文
- 与导师讨论方案

已创建项目看板
已设置里程碑提醒
已同步到日历

实验记录：

你：帮我记录今天的实验结果

OpenClaw：好的，请告诉我实验详情

你：实验3：使用Dropout优化模型
训练集准确率：92%
验证集准确率：89%
测试集准确率：87%

OpenClaw：已记录

【实验记录】实验3

📅 实验时间：2026-02-10
🎯 实验目的：使用Dropout优化模型，解决过拟合问题

⚙️ 实验配置：
- 模型：Transformer + Dropout(0.3)
- 数据集：IMDB
- Batch Size：32
- Learning Rate：0.001
- Epochs：20

📊 实验结果：
- 训练集准确率：92%
- 验证集准确率：89%
- 测试集准确率：87%

📈 对比分析：

模型	训练集	验证集	测试集
基线	95%	82%	80%
+正则化	93%	85%	83%
+Dropout	92%	89%	87%

💡 结论：
- Dropout有效缓解了过拟合
- 测试集准确率提升7%
- 建议继续优化超参数

📝 下一步：
- 尝试不同的Dropout率

```
- 结合数据增强
- 调整学习率

已保存到：实验记录/实验3.md
已更新项目进度
已生成可视化图表
```

11.4.7 完整工作流配置

学生学习助手配置脚本：

```
#!/bin/bash
# 学生学习助手完整配置

# 1. 课程资料管理
clawhub install file-organizer
openclaw config set courses.path "~/课程资料"
openclaw config set courses.auto-organize true

# 2. 论文阅读
clawhub install paper-reader
openclaw config set paper.language "中英文"
openclaw config set paper.notes-template "标准模板"

# 3. 作业辅助
clawhub install code-helper
openclaw config set homework.check true
openclaw config set homework.optimize true

# 4. 考试复习
clawhub install study-planner
clawhub install flashcard-generator
openclaw config set study.daily-reminder true

# 5. 科研项目
clawhub install project-manager
openclaw config set project.progress-tracking true
openclaw config set project.milestone-alert true

# 6. 时间管理
clawhub install calendar-sync
openclaw config set calendar.auto-create true

echo "✅ 学生学习助手配置完成！"
```text
### 11.4.8 学习效率提升数据

**使用前后对比**：

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 论文阅读 | 120分钟 | 30分钟 | 90分钟 | 75% |
| 笔记整理 | 45分钟 | 10分钟 | 35分钟 | 77.8% |
```
```

```
| 作业完成 | 180分钟 | 90分钟 | 90分钟 | 50% |
| 复习准备 | 60分钟 | 15分钟 | 45分钟 | 75% |
| 项目管理 | 30分钟 | 5分钟 | 25分钟 | 83.3% |
| **每日总计** | **435分钟** | **150分钟** | **285分钟** | **65.5%** |

**学期效率提升**：
```

节省时间：285分钟/天 × 100天 = 28,500分钟 ≈ 475小时
相当于：59个工作日
年度节省：118个工作日（约6个月）

```
**学习成果提升**：
```

使用前：
- 论文阅读：20篇/学期
- 平均成绩：85分
- 项目进度：经常延期

使用后：
- 论文阅读：50篇/学期（+150%）
- 平均成绩：92分（+8.2%）
- 项目进度：按时完成
- 额外收获：发表1篇论文

```
---

## 11.5 云上OpenClaw的5种正确打开方式

> 💡 **超级个体定位**：一个人 + OpenClaw = 无限可能。本节展示云端OpenClaw的5大实战场景，让你的AI助手24小时不间断工作。

### 11.5.1 场景概述

**云端OpenClaw的优势**：
```

✅ 24小时在线：不用开电脑
✅ 多端访问：手机、平板随时用
✅ 自动执行：定时任务自动跑
✅ 成本低廉：0.01元/月起步
✅ 快速部署：5分钟完成配置

百度云O计划部署：

![百度云OpenClaw 配置]
(https://upload.maynor1024.live/file/1770780868861_image_1.jpg)

配置步骤：
1. 选择OpenClaw应用镜像
2. 选择香港地区
3. 配置模型（ERNIE/DeepSeek/Qwen）
4. 配置消息平台（飞书/企微/钉钉/QQ）
5. 配置Skills（默认百度搜索）

![百度云O计划界面](https://upload.maynor1024.live/file/1770780895995_image_2.jpg)

成本对比：

配置	价格	适用场景
2核4G4M	0.01元/首月	轻量使用
4核8G8M	50元/月	中度使用
8核16G16M	150元/月	重度使用

11.5.2 场景1：智能日报系统

核心功能：定时抓取无RSS订阅的网站，生成个性化日报

配置方法：

```bash
# 1. 安装Skills双幻神

# 方法1：使用ClawHub安装（推荐）
npx clawhub@latest install find-skills
npx clawhub@latest install proactive-agent

# 方法2：通过对话安装
你：帮我安装这里面的Skills：
https://github.com/vercel-labs/skills/tree/main/skills/find-skills
https://github.com/leomariga/ProactiveAgent
```

```
OpenClaw：正在安装...
✅ find-skills 安装成功
✅ ProactiveAgent 安装成功

# 2. 配置定时任务
你：每天早上9点，搜索以下内容并生成日报：
1. AI行业最新动态（百度搜索）
2. OpenClaw社区更新（GitHub）
3. 技术博客精选（自定义RSS源）
4. 学术论文速递（百度学术）

格式要求：
- Markdown格式
- 分类整理
- 附带原文链接
- 推送到飞书
```text
**Skills双幻神介绍**：

**find-skills**：智能Skills发现
- 功能：OpenClaw遇到问题时主动寻找合适的Skills
- 优势：按需安装，避免Skills冗余
- 安装：`npx clawhub@latest install find-skills`
- GitHub：https://github.com/vercel-labs/skills/tree/main/skills/find-skills

**ProactiveAgent**：主动式Agent
- 功能：预测用户需求，主动发起对话
- 实战案例：做了几次日报转HTML后，主动提示"要不要我帮你自动化这个流程？"
- 安装：`npx clawhub@latest install proactive-agent`
- GitHub：https://github.com/leomariga/ProactiveAgent

⚠️ **安全提示**：ProactiveAgent安装时可能会显示VirusTotal警告（因为包含外部API调用），这是正常的，可以安全使用。

![百度Skills列表](https://upload.maynor1024.live/file/1770780916129_image_5.jpg)

**实战效果**：
```

【AI行业日报】2026-02-11

📊 今日要闻：

- OpenAI发布GPT-5.3，性能提升40%
```

- 国产大模型价格战升级，最低0.0001元/千tokens
- Anthropic推出Claude 5，支持100万tokens上下文

🔥 OpenClaw动态：
- 新增Skills：YouSkill搜索引擎集成
- 社区贡献：本周新增15个Skills
- Bug修复：修复飞书消息推送延迟问题

📚 技术博客：
- 《如何用OpenClaw打造个人知识库》
- 《Skills开发最佳实践》
- 《云端OpenClaw性能优化指南》

🎓 学术论文：
- Transformer优化新方法（ICLR 2026）
- 多模态大模型最新进展
- Agent系统架构设计

已推送到飞书 ✅

```
**避坑指南**：
- ⚠️ Brave Search需要绑卡，建议使用百度搜索（云端自带）
- ⚠️ 定时任务时区设置要正确
- ⚠️ 推送渠道要提前配置好

![Brave Search需要绑卡]
(https://upload.maynor1024.live/file/1770780906934_image_4.jpg)

---

### 11.5.3 场景2：智能备忘录管理

**痛点解决**：
- 传统备忘录：信息堆积，重要内容被淹没
- 时间轴管理：被动整理，效率低下
- AI摘要：单条处理，无法去重

**OpenClaw方案**：

```bash
# 配置每周自动整理
你：每周日晚上8点：
1. 读取本周所有备忘录
2. 智能去重（相同主题合并）
3. 分类整理（工作/学习/生活）
4. 提取未完成待办
5. 生成下周计划
6. 推送到飞书
```text
**实战效果**：
```

【本周备忘录整理】2026-02-10

📊 本周统计：
- 新增备忘录：45条
- 去重后：28条
- 重要事项：8条
- 已完成：20条
- 待处理：8条

📋 分类整理：

【工作】（15条）
✅ 完成客户A项目方案（已完成）
✅ 参加技术评审会（已完成）
⏳ 准备下周客户提案（进行中）
⏳ 完成项目文档（待开始）

【学习】（8条）
✅ 阅读React 19文档（已完成）
✅ 完成机器学习作业3（已完成）
⏳ 学习Transformer论文（进行中）

【生活】（5条）
✅ 预约体检（已完成）
⏳ 购买生日礼物（待处理）
⏳ 整理照片（待处理）

🎯 下周计划：
- 完成客户提案并演示
- 开始毕业论文实验
- 完成Transformer论文阅读
- 购买生日礼物

⚠️ 重要提醒：
- 客户提案截止时间：周三
- 生日礼物要提前准备
- 体检报告周二可取

已推送到飞书 ✅

```
![备忘录管理示例](https://upload.maynor1024.live/file/1770780930864_image_8.jpg)

---

### 11.5.4 场景3：滚动式待办提醒

**核心优势**：
- 在常用通讯工具中提醒（飞书/企微/钉钉）
- 提醒方式和频率无限制
- 结合日报自动更新

**配置方法**：

```bash
# 1. 每日待办生成
你：每天生成日报时，顺便：
1. 检查今日待办完成情况
2. 生成明日待办清单
3. 标注优先级
4. 设置提醒时间
5. 推送到飞书

# 2. 定时提醒
你：每天下午3点：
- 提醒未完成的高优先级任务
```text
**实战效果**：
```

【今日待办】2026-02-11 09:00

🎯 高优先级（3个）：
- 完成客户B报告终稿（截止：今天18:00）
- 参加技术评审会（时间：14:00）
- 提交项目周报（截止：今天17:00）

📋 中优先级（2个）：
- 阅读Transformer论文
- 整理会议纪要

🔄 低优先级（1个）：
- 更新个人博客

【下午提醒】2026-02-11 15:00

⚠️ 高优先级任务提醒：

・ 完成客户B报告终稿

- 截止时间：今天18:00
- 剩余时间：3小时
- 当前进度：80%
- 建议：抓紧完成

・ 提交项目周报

- 截止时间：今天17:00
- 剩余时间：2小时
- 当前进度：未开始
- 建议：立即开始

已@你 请及时处理 ⏰

![日报+待办事项示例](https://upload.maynor1024.live/file/1770780940542_image_9.jpg)

11.5.5 场景4：云端内容创作工作流

应用场景：PPT生成 + 素材管理 + 自动清理

工作流程：

```bash
# 1. 生成PPT
你：通过百度搜索和百度学术，生成一份介绍OpenClaw原理的PPT

OpenClaw：正在生成...

【PPT生成进度】
✅ 搜索相关资料（百度搜索）
✅ 查找学术论文（百度学术）
✅ 下载配图素材（20张）
✅ 生成PPT大纲
✅ 填充内容
✅ 设计排版
✅ 导出PPT

【生成结果】
• 文件名：OpenClaw原理介绍.pptx
• 页数：25页
• 素材：20张图片，5个图表
• 大小：15MB
• 下载链接：[点击下载]

素材文件已保存到：/output/openclaw-ppt/
```text
![PPT生成示例](https://upload.maynor1024.live/file/1770780875363_image_10.jpg)

**自动清理机制**：

```bash
# 2. 配置自动清理
你：每天早上9点，检查我的输出目录：
- 对比24小时前的备份
```

- 识别可删除的多余文件
- 生成清理清单
- 在日报中确认
- 我确认后才能删除

OpenClaw：已配置自动清理任务

【清理建议】2026-02-12 09:00

📁 输出目录检查：

可删除文件（3个）：
• openclaw-ppt/素材/ （20张图片，50MB）
  - 创建时间：昨天10:00
  - 最后访问：昨天10:30
  - 建议：PPT已下载，素材可删除

• temp-images/ （15张图片，30MB）
  - 创建时间：3天前
  - 最后访问：3天前
  - 建议：临时文件，可删除

• backup-20260210/ （备份文件，100MB）
  - 创建时间：2天前
  - 建议：已有新备份，旧备份可删除

💾 可释放空间：180MB

是否确认删除？回复"确认"执行删除
```text
**存储优化效果**：
- 2核4G4M配置完全够用
- 自动清理机制释放空间
- 云端处理，本地无负担

---

### 11.5.6 场景5：AI编程协作

**核心理念**：
- OpenClaw：规划和调度
```

- Claude Code：执行编程任务
- 成本优化：购买各家coding plan

工作流程：

```bash
# 1. 用户提出需求
你：帮我开发一个简单的待办事项Web应用

# 2. OpenClaw分析需求
OpenClaw：好的，让我分析一下需求...

【需求分析】
• 功能：待办事项的增删改查
• 技术栈：React + Node.js + MongoDB
• 预计工时：4小时
• 难度：中等

【开发计划】
1. 设计数据库结构（30分钟）
2. 开发后端API（90分钟）
3. 开发前端界面（120分钟）
4. 测试和调试（30分钟）

是否开始开发？

# 3. 调用Claude Code执行
你：开始

OpenClaw：正在调用Claude Code...

【开发进度】
✅ 数据库设计完成
✅ 后端API开发完成
✅ 前端界面开发完成
⏳ 测试中...

【代码统计】
• 后端代码：500行
• 前端代码：800行
• 测试代码：200行
• 总计：1500行
```

【项目结构】
todo-app/
├── backend/
│   ├── models/
│   ├── routes/
│   └── server.js
├── frontend/
│   ├── src/
│   └── public/
└── tests/

✅ 开发完成！
项目已保存到：~/projects/todo-app/
```text
![AI编程Skills](https://upload.maynor1024.live/file/1770780886607_image_11.jpg)

**成本优化**：
- OpenClaw：规划和管理（免费/低成本）
- Claude Code：编程执行（按需付费）
- 避免重复运行npx指令（节省token）
- 购买各家coding plan控制成本

---

### 11.5.7 人设与记忆管理技巧

**口头禅检测法**：

**问题**：长时间对话后OpenClaw可能遗忘设定

**解决方案**：

```json
{
  "systemPrompt": "你是我的AI助理。每次回复后，都要加上口头禅：'主人，还有什么需要帮忙的吗？'"
}
```text
![人设设置示例](https://upload.maynor1024.live/file/1770780919379_image_6.jpg)

**检测机制**：
- 一旦OpenClaw忘记加口头禅
- 立即发现记忆丢失
- 及时重新设定
```

```
**让OpenClaw教你配置**：

**最佳实践**：
- 不要死记硬背配置文档
- 把官方文档发给OpenClaw
- 让它教你如何配置

**实战案例**：
```

你：帮我配置飞书机器人
[附上飞书配置文档链接]

OpenClaw：好的，我来帮你配置。根据文档，需要以下步骤：

1. 在飞书开放平台创建应用
   - 访问：https://open.feishu.cn
   - 点击"创建企业自建应用"
   - 填写应用信息
2. 获取App ID和App Secret
   - 在应用详情页找到
   - 复制保存
3. 配置事件订阅
   - 设置请求地址
   - 订阅消息事件
4. 配置 OpenClaw

   ```
   {
     "feishu": {
       "appId": "你的App ID",
       "appSecret": "你的App Secret"
     }
   }
   ```text
   ```

需要我一步步指导你吗？
   ```

11.5.8 Skills生态补充资源

百度Skills库：

官方文档：https://cloud.baidu.com/doc/qianfan/s/Mmlda41a2

推荐Skills：
- baidu-search：百度搜索（云端自带）
- baidu-scholar：百度学术
- baidu-baike：百度百科
- 其他百度生态Skills

YouSkill - Skills搜索引擎：

网址：https://labs.youware.com/youskill

功能：
- 比find-skills更强大
- 提供多个Skills的组合工作流方案
- 社区驱动，想象力无穷

使用场景：
- 复杂任务需要多个Skills配合
- 寻找最佳Skills组合
- 学习社区最佳实践

实战演示：

```text
你：帮我在YouSkill上搜索"自动化日报"的解决方案

OpenClaw：正在搜索...

【YouSkill搜索结果】

方案1：基础日报系统
• Skills组合：
  - brave-search（联网搜索）
  - rss-reader（RSS订阅）
```

```
  - markdown-generator（格式化）
  - feishu-bot（推送）
• 适用场景：个人日报
• 难度：⭐⭐

方案2：智能日报系统
• Skills组合：
  - find-skills（智能发现）
  - ProactiveAgent（主动预测）
  - baidu-search（百度搜索）
  - content-analyzer（内容分析）
  - feishu-bot（推送）
• 适用场景：专业日报
• 难度：⭐⭐⭐

方案3：企业级日报系统
• Skills组合：
  - 方案2的所有Skills
  - database-connector（数据库）
  - analytics-dashboard（数据分析）
  - multi-channel-publisher（多渠道推送）
• 适用场景：团队日报
• 难度：⭐⭐⭐⭐

推荐：方案2（智能日报系统）
理由：功能完善，难度适中，适合个人使用
```

11.5.9 完整配置脚本

云上OpenClaw超级个体配置：

```bash
#!/bin/bash
# 云上OpenClaw超级个体完整配置

echo "🚀 开始配置云上OpenClaw..."

# 1. 安装Skills双幻神
echo "📦 安装Skills双幻神..."
npx clawhub@latest install find-skills
npx clawhub@latest install proactive-agent

# 2. 配置智能日报系统
echo "📰 配置智能日报系统..."
openclaw schedule add "daily-report" \
  --time "09:00" \
  --prompt "生成AI行业日报，包括：百度搜索、GitHub动态、技术博客、学术论文" \
  --channel "feishu"

# 3. 配置智能备忘录管理
echo "📝 配置智能备忘录管理..."
openclaw schedule add "weekly-notes-cleanup" \
  --time "Sun 20:00" \
  --prompt "整理本周备忘录：去重、分类、提取待办、生成下周计划"

# 4. 配置滚动式待办提醒
echo "⏰ 配置滚动式待办提醒..."
openclaw schedule add "daily-todo" \
  --time "09:00" \
  --prompt "生成今日待办清单"

openclaw schedule add "todo-reminder" \
  --time "15:00" \
  --prompt "提醒未完成的高优先级任务"

# 5. 配置云端内容创作
echo "🎨 配置云端内容创作..."
clawhub install banana-ai
openclaw config set content.auto-cleanup true

openclaw schedule add "file-cleanup" \
  --time "09:00" \
  --prompt "检查输出目录，生成清理建议"
```

```
# 6. 配置AI编程协作
echo "💻 配置AI编程协作..."
clawhub install code-planner
openclaw config set coding.use-claude-code true

# 7. 配置人设
echo "🤖 配置人设..."
openclaw config set systemPrompt "你是我的AI助理，专注于提升个人效率。每次回复后加上：'还有什么需要帮忙的吗？'"

# 8. 配置百度Skills
echo "🔍 配置百度Skills..."
openclaw config set search.engine "baidu"
openclaw config set search.scholar true

echo "✅ 云上OpenClaw 配置完成！"
echo ""
echo "📊 已配置功能："
echo "  ✅ 智能日报系统（每天9:00）"
echo "  ✅ 智能备忘录管理（每周日20:00）"
echo "  ✅ 滚动式待办提醒（每天9:00和15:00）"
echo "  ✅ 云端内容创作（自动清理）"
echo "  ✅ AI编程协作"
echo "  ✅ Skills双幻神"
echo ""
echo "🎯 你现在是一个超级个体了！"
```
```text
---

### 11.5.10 效率提升数据

**云上OpenClaw vs 本地OpenClaw**：

| 特性 | 本地OpenClaw | 云上OpenClaw | 优势 |
|------|-------------|-------------|------|
| 24小时在线 | ❌ | ✅ | +100% |
| 定时任务 | 需要开机 | 自动执行 | +100% |
| 多端访问 | 仅本机 | 手机/平板 | +200% |
| 存储管理 | 手动 | 自动清理 | +80% |
| 成本 | 电费+硬件 | 0.01元/月起 | -90% |
```

```
**超级个体效率提升**：
```

使用前（传统方式）：
- 每天手动查看新闻：30分钟
- 每周整理备忘录：60分钟
- 每天管理待办：20分钟
- 制作PPT：120分钟
- 编程项目规划：60分钟
- 总计：290分钟/天

使用后（云上OpenClaw）：
- 自动日报推送：0分钟（自动）
- 自动备忘录整理：0分钟（自动）
- 自动待办提醒：0分钟（自动）
- AI辅助PPT：20分钟
- AI编程协作：15分钟
- 总计：35分钟/天

节省时间：255分钟/天 ≈ 4.25小时/天
月度节省：94小时 ≈ 12个工作日
年度节省：1,128小时 ≈ 141个工作日

```
**投资回报率（ROI）**：
```

成本：

- 云服务器：0.01元/首月，50元/月（长期）
- Skills 配置：0元（开源免费）
- 学习成本：4小时
- 总成本：50元/月

收益：

- 节省时间：94小时/月
- 按时薪50元计算：4,700元/月
- ROI：(4,700 - 50) / 50 = 9,300%

```
---

## 📝 本章小结

通过4个真实场景 + 云上OpenClaw的5种应用，学习了如何用OpenClaw提升个人效率：

### 11.1 知识工作者
- 日报推送：节省30分钟/天
- 资料收集：效率提升96.7%
- 会议纪要：节省42分钟/会议
- 报告撰写：效率提升75%
- **总体提升**：81.2%

### 11.2 程序员
- 代码搜索：效率提升86.7%
- 文档查找：效率提升90%
- Bug追踪：效率提升75%
- 环境管理：效率提升90%
- **总体提升**：84.4%

### 11.3 内容创作者
- 选题策划：效率提升83.3%
- 资料收集：效率提升83.3%
- 内容创作：效率提升66.7%
- 多平台发布：效率提升66.7%
- **总体提升**：75.3%

### 11.4 学生
- 论文阅读：效率提升75%
- 作业完成：效率提升50%
- 考试复习：效率提升75%
- 项目管理：效率提升83.3%
- **总体提升**：65.5%

### 11.5 云上OpenClaw的5种应用
- 智能日报系统：24小时自动抓取
- 智能备忘录管理：自动去重分类
- 滚动式待办提醒：通讯工具中提醒
- 云端内容创作：PPT生成+自动清理
- AI编程协作：OpenClaw规划+Claude Code执行
- **Skills双幻神**：find-skills + ProactiveAgent
- **成本优势**：0.01元/月起步
```

核心要点

1. 自动化重复任务

✅ 日报推送
✅ 资料收集
✅ 会议纪要
✅ 数据分析

2. 系统化知识管理

✅ 统一存储
✅ 自动归档
✅ 智能搜索
✅ 知识图谱

3. 智能化辅助决策

✅ 选题建议
✅ 优化建议
✅ 风险提示
✅ 数据分析

4. 多平台协同

✅ 飞书/企微/钉钉
✅ Notion/备忘录
✅ 日历/提醒
✅ 多端同步

投资回报率（ROI）

成本：
- 配置时间：4-8小时
- API费用：$30-100/月
- 学习成本：1-2周

收益：
- 时间节省：65%-85%
- 质量提升：20%-50%
- 压力降低：显著
- 成长加速：明显

ROI：5,000%-10,000%

成功关键

1. **明确目标**：知道自己要解决什么问题
2. **循序渐进**：从简单场景开始
3. **持续优化**：根据使用情况调整
4. **形成习惯**：让OpenClaw成为日常工具

💡 思考题

1. 你的工作/学习中，哪些任务最耗时？
2. 这些任务中，哪些可以用OpenClaw自动化？
3. 如何设计适合自己的工作流？
4. 如何衡量效率提升的效果？

下一章预告：第12章将学习个人效率进阶，包括高级自动化工作流、多Skills组合应用、个人知识图谱构建和效率优化策略。

11.2.9 Coding Agent：让OpenClaw操控Claude Code

```
### 11.2.9.1 场景描述

**核心理念**：
- OpenClaw：规划和调度
- Claude Code：执行编程任务
- 两者配合：实现全自动开发

**工作流程**：
```

需求 → OpenClaw分析 → 生成开发计划 →
调用Claude Code → 自动编程 → 测试验证 →
发现Bug → 自动修复 → 完成交付

```
**优势**：
```

✅ 24小时不间断开发
✅ 自动测试和修复
✅ 代码质量稳定
✅ 开发效率提升10倍

```
### 11.2.9.2 安装Claude Code

**方式1：让OpenClaw自动安装**
```

你：帮我安装Claude Code

OpenClaw：好的，正在安装Claude Code...

📦 【安装Claude Code】

⏳ 安装步骤：

1. 检查系统环境... ✅
2. 下载Claude Code... ✅
3. 安装依赖... ✅
4. 配置环境变量... ✅
5. 测试安装... ✅

✅ 安装完成！

现在可以开始使用了！

```
![Claude Code安装](https://upload.maynor1024.live/file/1770782761573_image_28.jpg)

**方式2：手动安装**

```bash
# 1. 安装Claude CLI
npm install -g @anthropic-ai/claude-cli

# 2. 配置API Key
export ANTHROPIC_API_KEY="your-api-key"

# 3. 配置 OpenClaw
clawhub install coding-agent
openclaw config set coding.tool "claude-code"
```text
### 11.2.9.3 基础使用：简单开发任务

**场景：开发一个待办事项应用**
```

你：用Claude Code帮我开发一个待办事项应用，
要求：
- 使用React + TypeScript
- 支持添加、删除、完成任务
- 数据持久化到localStorage
- 界面简洁美观

OpenClaw：好的，正在调用Claude Code开发...

🔧 【开发任务】待办事项应用

步骤1：创建项目结构 ✅
步骤2：实现核心功能 ✅
步骤3：添加样式 ✅
步骤4：测试功能 ✅

✅ 开发完成！

📁 项目文件：
• 源代码：todo-app/
• 打包文件：todo-app/dist/

🚀 运行方法：
cd todo-app

npm install
npm run dev

已发送项目文件到飞书 ✅

```
![Claude Code开发示例]
(https://upload.maynor1024.live/file/1770782764980_image_29.jpg)

### 11.2.9.4 进阶使用：自动测试和修复

**场景：开发过程中自动发现和修复Bug**
```

你：用Claude Code开发一个用户管理系统，
要求自动测试并修复所有Bug

OpenClaw：好的，正在开发并测试...

阶段1：后端开发 ✅
阶段2：前端开发 ✅
阶段3：自动测试 ⏳

测试结果：
✅ 通过：45个测试
❌ 失败：3个测试

阶段4：自动修复Bug ⏳
正在修复Bug...

修复1：邮箱格式验证 ✅
修复2：级联删除 ✅
修复3：分页计算 ✅

阶段5：回归测试 ✅
✅ 通过：48个测试
❌ 失败：0个测试

✅ 全部测试通过！

```
### 11.2.9.5 配置方法

```bash
# 1. 安装Coding Agent Skill
clawhub install coding-agent

# 2. 配置Claude Code
openclaw config set coding.tool "claude-code"
openclaw config set coding.model "claude-3-5-sonnet"
openclaw config set coding.api-key "YOUR_ANTHROPIC_API_KEY"

# 3. 配置开发环境
openclaw config set coding.workspace "~/projects"
openclaw config set coding.auto-test true
openclaw config set coding.auto-fix true
```text
### 11.2.9.6 效率提升数据

**使用Coding Agent前后对比**：

| 任务类型 | 传统开发 | Coding Agent | 节省时间 | 提升比例 |
|---------|---------|--------------|----------|----------|
| 简单功能 | 2小时 | 10分钟 | 110分钟 | 91.7% |
| 中等项目 | 8小时 | 45分钟 | 435分钟 | 90.6% |
| 大型项目 | 40小时 | 2小时 | 2,280分钟 | 95% |
| Bug修复 | 1小时 | 5分钟 | 55分钟 | 91.7% |
| **平均** | **51.5小时** | **3.5小时** | **2,908分钟** | **93.2%** |

**程序员效率提升**：
```

每周开发任务：
- 新功能开发：3个 × 8小时 = 24小时
- Bug修复：5个 × 1小时 = 5小时
- 总计：29小时

使用Coding Agent后：
- 新功能开发：3个 × 45分钟 = 2.25小时
- Bug修复：5个 × 5分钟 = 0.42小时
- 总计：2.67小时

每周节省：26.33小时
每月节省：105小时 ≈ 13个工作日

```
### 11.2.9.7 注意事项

**安全性**：
```

⚠️ 重要提醒：

1. 代码审查：AI生成的代码需要人工审查
2. 测试验证：必须进行充分测试
3. 安全检查：检查是否有安全漏洞
4. 备份代码：定期备份重要代码

```
**成本控制**：
```

💰 成本优化：

1. 使用Coding Plan：Claude Pro $20/月
2. 合理使用：避免过度依赖
3. 缓存结果：相似任务复用代码
4. 批量处理：一次性完成多个任务

```
---

## 相关资源

- Claude Code官网：https://claude.ai/code
- Anthropic API文档：https://docs.anthropic.com
- Coding Agent Skill：https://clawhub.ai/skills/coding-agent
```

第13章 高级自动化工作流（多Skills组合/知识图谱）

> 💡 **本章目标**：学习高级自动化工作流、多Skills组合应用、个人知识图谱构建和效率优化策略，让你成为真正的超级个体。

🎯 本章内容

- 13.1 高级自动化工作流
- 13.2 多Skills组合应用
- 13.3 个人知识图谱构建
- 13.4 效率优化策略
- 13.5 5个高级自动化案例（ROI 9000%-15900%）
- 12.5 Coding Agent工作流

12.1 高级自动化工作流

> 💡 **核心价值**：通过自动化工作流，将重复性工作交给OpenClaw，你只需要专注于创造性工作。

12.1.1 全自动信息收集系统

场景描述

作为超级个体，你需要持续学习和获取信息，但手动收集信息太耗时。通过OpenClaw构建全自动信息收集系统，每天早上自动生成个性化日报。

工作流程：

```
多源信息收集 → 智能去重 → 质量评分 → 自动分类 → 生成日报 → 飞书推送
```

实战配置

第一步：安装必需的Skills

```
# 安装信息收集Skills
clawhub install brave-search      # 网页搜索
clawhub install rss-reader        # RSS订阅
clawhub install github-trending   # GitHub热门

# 安装内容处理Skills
clawhub install content-analyzer  # 内容分析
clawhub install text-summarizer   # 文本摘要
clawhub install duplicate-checker # 去重检查
```text
**第二步：配置信息源**

创建配置文件 `~/.openclaw/info-sources.json`：

```json
{
  "sources": {
    "rss": [
      {
        "name": "阮一峰的网络日志",
        "url": "https://www.ruanyifeng.com/blog/atom.xml",
        "category": "技术"
      },
      {
        "name": "少数派",
        "url": "https://sspai.com/feed",
        "category": "效率"
      }
    ],
    "github": {
      "trending": {
        "language": "python",
        "since": "daily"
      },
      "repos": [
        "openclaw/openclaw",
        "microsoft/vscode"
      ]
    },
    "keywords": [
      "OpenClaw",
      "AI工具",
```

```
      "效率提升",
      "自动化"
    ]
  },
  "filter": {
    "keywords": ["AI", "自动化", "效率", "工具"],
    "exclude": ["广告", "营销", "推广"],
    "quality_threshold": 75,
    "max_items": 20
  },
  "schedule": {
    "time": "08:00",
    "timezone": "Asia/Shanghai"
  }
}
```text
**第三步：创建自动化脚本**

创建 `~/.openclaw/scripts/daily-digest.sh`：

```bash
#!/bin/bash

# 每日信息收集脚本
DATE=$(date +%Y-%m-%d)
OUTPUT_DIR="$HOME/.openclaw/digests"
mkdir -p "$OUTPUT_DIR"

# 1. 收集RSS信息
echo "📰 收集RSS信息..."
openclaw agent --message "请使用 rss-reader skill 收集 ~/.openclaw/info-sources.json
中配置的RSS源，保存到 $OUTPUT_DIR/rss-$DATE.json"

# 2. 收集GitHub热门
echo "🔥 收集GitHub热门..."
openclaw agent --message "请收集GitHub今日Python热门项目，保存到
$OUTPUT_DIR/github-$DATE.json"

# 3. 搜索关键词
echo "🔍 搜索关键词..."
openclaw agent --message "请搜索'OpenClaw AI工具'相关信息，最多10条结果，保存到
$OUTPUT_DIR/search-$DATE.json"
```

```
# 4. 合并和去重
echo "🔄 合并和去重..."
openclaw agent --message "请合并 $OUTPUT_DIR/*-$DATE.json 中的所有信息并去重，保存到
$OUTPUT_DIR/merged-$DATE.json"

# 5. 内容分析和评分
echo "📊 内容分析..."
openclaw agent --message "请分析 $OUTPUT_DIR/merged-$DATE.json 中的内容并评分，保存到
$OUTPUT_DIR/analyzed-$DATE.json"

# 6. 生成日报
echo "⌧ 生成日报..."
openclaw agent --message "请根据以下信息生成今日日报，按技术、产品、行业分类，每条信息包含标
题、摘要、链接和推荐理由：
$(cat $OUTPUT_DIR/analyzed-$DATE.json)" \
  --output "$OUTPUT_DIR/digest-$DATE.md"

# 7. 推送到飞书
echo "⌧ 推送到飞书..."
openclaw channels send feishu \
  --message "$(cat $OUTPUT_DIR/digest-$DATE.md)" \
  --title "📰 每日资讯 $DATE"

echo "✅ 日报生成完成！"
```text
**第四步：配置定时任务**

```bash
# 添加到crontab
crontab -e

# 每天早上8点执行
0 8 * * * /bin/bash ~/.openclaw/scripts/daily-digest.sh
```text
#### 实战案例：技术资讯自动收集

**场景**：作为开发者，需要每天了解最新的技术动态。

**配置示例**：

```json
{
  "sources": {
```

```
    "rss": [
      "https://news.ycombinator.com/rss",
      "https://www.reddit.com/r/programming/.rss",
      "https://dev.to/feed"
    ],
    "github": {
      "trending": ["python", "javascript", "go"],
      "topics": ["ai", "automation", "productivity"]
    },
    "twitter": {
      "accounts": ["@OpenAI", "@github", "@vercel"],
      "hashtags": ["#AI", "#DevTools"]
    }
  },
  "filter": {
    "keywords": ["AI", "开源", "工具", "框架"],
    "min_stars": 100,
    "quality_threshold": 80
  }
}
```text
**效果数据**：
- 每天自动收集50+条信息
- 智能过滤后保留15-20条高质量内容
- 节省时间：每天2小时
- 信息质量：提升60%

### 12.1.2 智能任务管理系统

#### 场景描述

传统的待办清单只是记录任务，智能任务管理系统能够：
- 自动识别任务优先级
- 智能安排执行时间
- 自动跟踪任务进度
- 主动提醒和推动

**工作流程**：
```

任务输入 → 智能分析 → 优先级排序 → 时间安排 → 进度跟踪 → 完成复盘
```

实战配置

第一步：安装任务管理Skills

```bash
clawhub install task-manager
clawhub install calendar-sync
clawhub install priority-analyzer
clawhub install time-estimator
```text
**第二步：配置任务规则**

创建 `~/.openclaw/task-rules.json`：

```json
{
  "priority_rules": {
    "urgent_keywords": ["紧急", "立即", "今天必须"],
    "important_keywords": ["重要", "关键", "核心"],
    "deadline_weight": 0.4,
    "impact_weight": 0.3,
    "effort_weight": 0.3
  },
  "time_rules": {
    "work_hours": "09:00-18:00",
    "focus_time": "09:00-11:00",
    "meeting_time": "14:00-16:00",
    "break_interval": 90
  },
  "automation": {
    "auto_schedule": true,
    "auto_reminder": true,
    "auto_followup": true
  }
}
```text
**第三步：创建任务管理对话**

在飞书中与OpenClaw对话：

你：帮我管理今天的任务
OpenClaw：好的，请告诉我你今天的任务清单

你：

1. 完成项目文档（预计2小时）
2. 代码review（预计1小时）
3. 团队会议（14:00-15:00）
4. 修复bug（紧急，预计3小时）
5. 学习新技术（预计1小时）

OpenClaw：已分析你的任务，建议执行顺序：

🔴 优先级1（紧急重要）

- 修复bug（09:00-12:00，3小时）
  理由：标记为紧急，安排在上午专注时间

🟡 优先级2（重要不紧急）

- 完成项目文档（13:00-14:00，1小时）
  理由：重要任务，会议前完成
- 代码review（15:00-16:00，1小时）
  理由：需要专注，会议后进行

🟢 优先级3（不紧急）

- 团队会议（14:00-15:00，固定）
- 学习新技术（16:00-17:00，1小时）
  理由：可灵活安排，放在下午

已同步到你的日历，并设置提醒。需要调整吗？

```
实战案例：项目管理自动化

场景：管理一个多任务项目，需要跟踪进度和协调资源。

配置示例：

```bash
# 创建项目
openclaw agent --message "创建项目：OpenClaw教程优化
任务列表：
1. 修复Docker文档（2小时，已完成）
2. 补充第13章内容（8小时，进行中）
3. 删除重复内容（4小时，待开始）
4. 统一代码格式（3小时，待开始）
5. 更新交叉引用（2小时，待开始）

请帮我制定执行计划"

# OpenClaw自动生成甘特图和执行计划
```text
**效果数据**：
- 任务完成率：从70% → 95%
- 时间利用率：提升40%
- 拖延减少：80%
- 压力降低：60%

### 12.1.3 自动化内容创作流程

#### 场景描述

内容创作是超级个体的核心能力，但从灵感到发布的流程很繁琐。通过自动化流程，大幅提升创作效率。

**工作流程**：
```

灵感收集 → 素材整理 → 大纲生成 → 内容创作 → 排版优化 → 多平台发布

实战配置

第一步：灵感收集系统

```bash
# 在飞书中随时记录灵感
你：灵感：OpenClaw可以用来做自动化测试
OpenClaw：已记录灵感到「内容创作」分类
- 标题：OpenClaw自动化测试应用
- 关键词：自动化、测试、OpenClaw
- 相关素材：已搜索3篇相关文章
- 建议大纲：已生成初步大纲
- 预计字数：2000-3000字

需要现在开始创作吗？
```text
**第二步：自动化创作脚本**

创建 `~/.openclaw/scripts/content-creation.sh`：

```bash
#!/bin/bash

TOPIC=$1
OUTPUT_DIR="$HOME/.openclaw/content"
mkdir -p "$OUTPUT_DIR"

# 1. 搜索相关资料
echo "🔍 搜索相关资料..."
openclaw skills run brave-search \
  --query "$TOPIC" \
  --max-results 10 \
  --output "$OUTPUT_DIR/research.json"

# 2. 生成大纲
echo "📋 生成大纲..."
openclaw agent --message "根据以下资料生成文章大纲：
主题：$TOPIC
资料：$(cat $OUTPUT_DIR/research.json)
要求：
- 结构清晰，3-5个章节
- 包含实战案例
```

```
- 字数2000-3000字" \
  --output "$OUTPUT_DIR/outline.md"

# 3. 创作内容
echo "✍️ 创作内容..."
openclaw agent --message "根据大纲创作完整文章：
$(cat $OUTPUT_DIR/outline.md)
要求：
- 语言通俗易懂
- 包含代码示例
- 添加配图建议" \
  --output "$OUTPUT_DIR/draft.md"

# 4. 优化排版
echo "🎨 优化排版..."
openclaw agent --message "优化文章排版：
$(cat $OUTPUT_DIR/draft.md)
要求：
- 添加emoji
- 优化标题层级
- 添加引用和提示框" \
  --output "$OUTPUT_DIR/final.md"

echo "✅ 内容创作完成！"
echo "📄 文件位置：$OUTPUT_DIR/final.md"
```text
**使用方法**：

```bash
# 创作一篇文章
bash ~/.openclaw/scripts/content-creation.sh "OpenClaw自动化测试实战"

# 10分钟后，文章创作完成
```text
#### 实战案例：技术博客自动化

**场景**：每周发布1-2篇技术博客，从选题到发布全流程自动化。

**完整流程**：

```bash
# 周一：生成本周选题
openclaw agent --message "分析最近的技术热点，生成3个博客选题"
```

```
# 周二：创作第一篇
bash ~/.openclaw/scripts/content-creation.sh "选题1"

# 周三：审核和优化
openclaw agent --message "审核文章，提出优化建议：
$(cat ~/.openclaw/content/final.md)"

# 周四：生成配图
openclaw agent --message "请生成技术博客配图，风格：简约科技风"

# 周五：多平台发布
openclaw agent --message "将文章发布到：
- 个人博客
- 掘金
- CSDN
- 知乎"
```text
**效果数据**：
- 创作时间：从8小时 → 2小时
- 发布频率：从月更 → 周更
- 内容质量：保持稳定
- 阅读量：提升50%

### 12.1.4 效率数据监控系统

#### 场景描述

要优化效率，首先要量化效率。通过数据监控系统，实时了解自己的工作状态。

**监控指标**：
- 时间分配：工作、学习、休息
- 任务完成率：计划vs实际
- 专注时长：深度工作时间
- 自动化覆盖率：自动化任务占比

#### 实战配置

**第一步：配置数据收集**

```json
{
  "tracking": {
```

```
    "time_tracking": {
      "enabled": true,
      "categories": ["工作", "学习", "休息", "娱乐"],
      "auto_detect": true
    },
    "task_tracking": {
      "enabled": true,
      "sync_calendar": true,
      "track_completion": true
    },
    "focus_tracking": {
      "enabled": true,
      "pomodoro": 25,
      "break": 5
    }
  },
  "reporting": {
    "daily_summary": "20:00",
    "weekly_review": "Sunday 18:00",
    "monthly_report": "Last day 18:00"
  }
}
```text
**第二步：自动生成效率报告**

```bash
# 每日总结
openclaw agent --message "生成今日效率报告"

# OpenClaw自动生成：
📊 今日效率报告 2026-02-11

⏰ 时间分配
- 工作：6.5小时（目标8小时）
- 学习：2小时
- 休息：1.5小时

✅ 任务完成
- 计划任务：8个
- 完成任务：7个
- 完成率：87.5%

🎯 专注时长
```

```
- 深度工作：4小时
- 番茄钟：8个
- 专注度：85%

🤖 自动化效果
- 自动化任务：12个
- 节省时间：2.5小时
- 自动化率：60%

💡 改进建议
1. 增加深度工作时间到5小时
2. 减少会议时间
3. 提高自动化覆盖率到70%
```text
**效果数据**：
- 时间利用率：提升35%
- 工作效率：提升50%
- 压力水平：降低40%
- 工作满意度：提升60%

---

## 12.2 多Skills组合应用

> 💡 **核心价值**：单个Skill是工具，多个Skills组合是系统。通过组合应用，实现1+1>2的效果。

### 12.2.1 Skills组合策略

#### 基础组合模式

**模式1：串行组合**
```

Skill A → Skill B → Skill C

```
适用场景：有明确的处理流程

**模式2：并行组合**
```

```
    → Skill A →
```

Input → → Skill B → → Output
→ Skill C →
```

```
适用场景：需要多角度处理

**模式3：条件组合**
```

Input → 判断 → Skill A（条件1）
→ Skill B（条件2）

```
适用场景：根据条件选择不同处理方式

#### 经典组合案例

**组合1：智能日报系统**
```

find-skills（发现新Skills）
↓
ProactiveAgent（预测需求）
↓
brave-search（搜索相关信息）
↓
content-analyzer（分析内容）
↓
markdown-generator（生成报告）

```
**组合2：知识管理系统**
```

web-clipper（网页剪藏）
↓
content-extractor（提取正文）
↓
text-summarizer（生成摘要）
↓
tag-generator（自动打标签）
↓
notion-sync（同步到Notion）

```
**组合3：代码学习系统**
```

github-search（搜索项目）
↓
code-analyzer（分析代码）
↓

dependency-checker（检查依赖）
↓
doc-generator（生成文档）
↓
knowledge-graph（构建知识图谱）

```
### 12.2.2 实战案例：全自动学习系统

#### 场景描述

作为超级个体，需要持续学习新技术。通过Skills组合，构建全自动学习系统。

**学习流程**：
```

发现学习资源 → 内容提取 → 知识整理 → 实践练习 → 复盘总结

实战配置

第一步：安装学习相关Skills

```bash
# 内容发现
clawhub install github-trending
clawhub install course-finder
clawhub install paper-search

# 内容处理
clawhub install pdf-reader
clawhub install video-transcriber
clawhub install note-taker

# 知识管理
clawhub install flashcard-generator
clawhub install mind-map-creator
clawhub install spaced-repetition
```text
**第二步：创建学习工作流**

创建 `~/.openclaw/workflows/learning.json`：

```json
{
  "workflow": "自动化学习系统",
  "steps": [
    {
      "name": "发现学习资源",
      "skills": ["github-trending", "course-finder"],
      "config": {
        "topics": ["AI", "自动化", "效率工具"],
        "quality_threshold": 80
      }
    },
    {
      "name": "内容提取",
      "skills": ["pdf-reader", "video-transcriber"],
      "config": {
        "extract_key_points": true,
        "generate_summary": true
```

```
      }
    },
    {
      "name": "知识整理",
      "skills": ["note-taker", "mind-map-creator"],
      "config": {
        "format": "markdown",
        "auto_categorize": true
      }
    },
    {
      "name": "记忆强化",
      "skills": ["flashcard-generator", "spaced-repetition"],
      "config": {
        "review_schedule": "1,3,7,15,30"
      }
    }
  ]
}
```text
**第三步：执行学习工作流**

```bash
# 启动学习工作流
openclaw workflow run learning

# 或通过对话启动
openclaw agent --message "我想学习Python异步编程，帮我制定学习计划"

# OpenClaw自动执行：
1. 搜索Python异步编程相关资源
   - GitHub热门项目：5个
   - 优质教程：3个
   - 技术文章：10篇

2. 提取核心知识点
   - async/await语法
   - 协程原理
   - 事件循环
   - 常用库：asyncio, aiohttp

3. 生成学习笔记
   - 概念解释
```

```text
    - 代码示例
    - 实践项目

4. 创建复习卡片
    - 生成20张闪卡
    - 设置复习计划

5. 推荐实践项目
    - 异步爬虫
    - 异步API服务
    - 异步任务队列
```
```text
#### 效果数据

**学习效率提升**：
- 资源发现时间：从2小时 → 10分钟
- 笔记整理时间：从1小时 → 5分钟
- 知识留存率：从40% → 80%
- 学习速度：提升3倍

### 12.2.3 实战案例：内容创作工作流

#### 场景描述

内容创作需要灵感、素材、创作、优化多个环节，通过Skills组合实现全流程自动化。

**创作流程**：
```

灵感收集 → 素材搜索 → 大纲生成 → 内容创作 → 图片生成 → 排版优化 → 多平台发布

Skills组合方案

```bash
# 灵感收集
idea-collector + trend-analyzer
  ↓
# 素材搜索
brave-search + content-scraper + image-search
  ↓
# 大纲生成
outline-generator + structure-optimizer
  ↓
# 内容创作
ai-writer + code-generator + example-creator
  ↓
# 图片生成
image-generator + image-optimizer
  ↓
# 排版优化
markdown-formatter + style-checker
  ↓
# 多平台发布
blog-publisher + social-media-poster
```text
#### 实战配置

```bash
# 创建内容创作工作流
openclaw agent --message "我要写一篇关于OpenClaw自动化的文章"

# OpenClaw自动执行：

📝 第1步：分析热点趋势
- 搜索"OpenClaw自动化"相关内容
- 分析热门话题和用户痛点
- 生成3个选题方向

📚 第2步：收集素材
- 搜索相关文章：15篇
- 提取优质案例：8个
- 收集配图素材：20张
```

```
📋 第3步：生成大纲
一、OpenClaw自动化概述
二、5个实战场景
三、配置方法详解
四、效果数据展示
五、避坑指南

✍️ 第4步：创作内容
- 自动生成各章节内容
- 插入代码示例
- 添加实战案例

🎨 第5步：生成配图
- 封面图：科技风格
- 流程图：3张
- 效果对比图：2张

🔧 第6步：优化排版
- 添加emoji
- 优化标题层级
- 添加引用框

📤 第7步：多平台发布
- 个人博客：已发布
- 掘金：已发布
- 知乎：已发布
- 公众号：草稿已生成

✅ 创作完成！总耗时：15分钟
```text
#### 效果对比

| 环节 | 传统方式 | Skills组合 | 效率提升 |
|------|----------|------------|----------|
| 选题 | 1小时 | 5分钟 | 12倍 |
| 素材收集 | 2小时 | 10分钟 | 12倍 |
| 大纲 | 30分钟 | 2分钟 | 15倍 |
| 创作 | 4小时 | 10分钟 | 24倍 |
| 配图 | 1小时 | 3分钟 | 20倍 |
| 发布 | 30分钟 | 2分钟 | 15倍 |
| **总计** | **9小时** | **32分钟** | **17倍** |

### 12.2.4 实战案例：数据分析工作流
```
```

```
#### 场景描述

数据分析需要收集、清洗、分析、可视化多个步骤，通过Skills组合实现自动化。

**分析流程**：
```

数据收集 → 数据清洗 → 数据分析 → 可视化 → 报告生成

Skills组合方案

```bash
# 数据收集
api-connector + web-scraper + database-query
  ↓
# 数据清洗
data-cleaner + duplicate-remover + format-converter
  ↓
# 数据分析
statistical-analyzer + trend-detector + anomaly-finder
  ↓
# 可视化
chart-generator + dashboard-creator
  ↓
# 报告生成
report-writer + insight-summarizer
```text
#### 实战配置

```bash
# 分析GitHub项目数据
openclaw agent --message "分析OpenClaw项目的增长趋势"

# OpenClaw自动执行：

📊 第1步：收集数据
- Star历史：1000条记录
- Fork历史：500条记录
- Issue统计：200个
- PR统计：150个

🧹 第2步：数据清洗
- 去除重复数据
- 填充缺失值
- 标准化格式

📈 第3步：数据分析
- 增长趋势：月均增长15%
- 活跃度：高峰期在工作日
- 用户画像：70%开发者
- 热门功能：Skills最受欢迎
```

```
📉 第4步：生成图表
- Star增长曲线
- 用户活跃度热力图
- 功能使用占比饼图
- 问题分类柱状图

📝 第5步：生成报告
《OpenClaw项目数据分析报告》
- 核心指标
- 增长趋势
- 用户洞察
- 改进建议

✅ 分析完成！
```text
#### 效果数据

- 分析时间：从4小时 → 10分钟
- 数据准确性：提升40%
- 洞察深度：提升60%
- 决策效率：提升80%

### 12.2.5 Skills组合最佳实践

#### 实践1：模块化设计

**原则**：每个Skill专注做好一件事

```bash
# 不好的做法：一个Skill做所有事
clawhub install all-in-one-tool

# 好的做法：多个专业Skill组合
clawhub install data-collector
clawhub install data-analyzer
clawhub install report-generator
```text
#### 实践2：错误处理

**原则**：每个环节都要有容错机制

```json
```

```
{
  "workflow": "内容创作",
  "error_handling": {
    "retry": 3,
    "fallback": "use_default",
    "notification": true
  },
  "steps": [
    {
      "name": "搜索素材",
      "skill": "brave-search",
      "on_error": "use_cached_data"
    },
    {
      "name": "生成内容",
      "skill": "ai-writer",
      "on_error": "use_template"
    }
  ]
}
```text
#### 实践3：性能优化

**原则**：并行执行可以并行的任务

```bash
# 串行执行（慢）
搜索资料 → 生成大纲 → 创作内容 → 生成配图
总耗时：20分钟

# 并行执行（快）
搜索资料 ─┬→ 生成大纲 → 创作内容
          └→ 生成配图
总耗时：12分钟
```text
#### 实践4：数据流转

**原则**：标准化数据格式，便于Skills之间传递

```json
{
  "data_format": {
    "input": {
```

```
      "type": "json",
      "schema": "standard_v1"
    },
    "output": {
      "type": "json",
      "schema": "standard_v1"
    }
  }
}
```text
### 12.2.6 避坑指南

#### 坑1：Skills冲突

**问题**：多个Skills同时修改同一数据

**解决**：
```bash
# 使用锁机制
openclaw config set skills.lock true

# 或串行执行
openclaw workflow run --mode sequential
```text
#### 坑2：资源消耗

**问题**：同时运行太多Skills导致系统卡顿

**解决**：
```bash
# 限制并发数
openclaw config set skills.max_concurrent 3

# 设置优先级
openclaw config set skills.priority '{
  "critical": ["task-manager"],
  "high": ["content-creator"],
  "normal": ["data-analyzer"]
}'
```text
#### 坑3：依赖问题

**问题**：Skill B依赖Skill A的输出，但A失败了
```
```

解决：
```json
{
  "dependencies": {
    "skill-b": {
      "requires": ["skill-a"],
      "on_missing": "skip"
    }
  }
}
```text
### 12.2.7 组合效果评估

#### 评估指标

**效率指标**：
- 时间节省率 = (原时间 - 新时间) / 原时间
- 自动化率 = 自动化任务数 / 总任务数

**质量指标**：
- 准确率 = 正确结果数 / 总结果数
- 完成率 = 完成任务数 / 计划任务数

**成本指标**：
- ROI = (收益 - 成本) / 成本
- 学习成本 = 配置时间 + 学习时间

#### 实际案例数据

**案例：内容创作工作流**
- 时间节省率：82%（9小时 → 1.5小时）
- 自动化率：90%
- 内容质量：保持稳定
- ROI：1500%（投入2小时配置，每周节省15小时）

**案例：数据分析工作流**
- 时间节省率：92%（4小时 → 20分钟）
- 准确率：98%
- 洞察深度：提升60%
- ROI：2000%

---

```
## 12.3 个人知识图谱构建

> 💡 **核心价值**：将碎片化知识系统化，构建个人知识体系，实现知识的积累和复用。

### 12.3.1 知识图谱基础

#### 什么是知识图谱

**定义**：知识图谱是一种结构化的知识表示方法，用节点表示实体（概念、人物、事件等），用边表示实体之间的关系。

**个人知识图谱的价值**：
```

碎片知识 → 结构化 → 关联化 → 可视化 → 可检索 → 可复用

```
**核心要素**：
- **节点（Node）**：知识点、概念、技能
- **边（Edge）**：关系、依赖、引用
- **属性（Property）**：标签、时间、来源

#### 知识图谱示例
```

OpenClaw（核心概念）
├── 包含 → Skills（子概念）
│ ├── 包含 → find-skills
│ ├── 包含 → ProactiveAgent
│ └── 包含 → brave-search
├── 应用于 → 自动化（应用场景）
│ ├── 包含 → 信息收集
│ ├── 包含 → 任务管理
│ └── 包含 → 内容创作
└── 相关 → AI工具（相关概念）
├── 包含 → ChatGPT
├── 包含 → Claude
└── 包含 → Copilot

12.3.2 构建个人知识图谱

第一步：知识收集

方法1：自动收集

```bash
# 从笔记中提取知识点
clawhub install note-parser
openclaw agent --message "请使用 note-parser skill 从 ~/.openclaw/notes 提取知识点，保存到 ~/.openclaw/knowledge/entities.json"

# 从浏览历史提取
clawhub install browser-history-analyzer
openclaw agent --message "请使用 browser-history-analyzer skill 分析最近30天的浏览历史，保存到 ~/.openclaw/knowledge/topics.json"

# 从对话记录提取
openclaw agent --message "分析我最近的对话，提取关键知识点"
```text
**方法2：手动标注**

```bash
# 在飞书中标注知识点
你：#知识点 OpenClaw可以通过Skills扩展功能
OpenClaw：已添加到知识图谱
- 实体：OpenClaw, Skills
- 关系：扩展功能
- 分类：技术/工具

你：#知识点 find-skills可以发现新的Skills
OpenClaw：已添加到知识图谱
- 实体：find-skills, Skills
- 关系：发现
- 关联：OpenClaw → Skills → find-skills
```text
#### 第二步：关系提取

**自动提取关系**：

```bash
# 分析知识点之间的关系
```

```
openclaw agent --message "分析我的知识库，提取知识点之间的关系"

# OpenClaw自动分析：
发现关系：
1. OpenClaw → 包含 → Skills（层级关系）
2. Skills → 依赖 → API配置（依赖关系）
3. find-skills → 相似 → ProactiveAgent（相似关系）
4. 自动化 → 应用 → OpenClaw（应用关系）
5. Docker → 部署方式 → OpenClaw（方式关系）

已更新知识图谱
```text
**手动定义关系**：

```json
{
  "relationships": [
    {
      "from": "OpenClaw",
      "to": "Skills",
      "type": "包含",
      "weight": 1.0
    },
    {
      "from": "Skills",
      "to": "自动化",
      "type": "实现",
      "weight": 0.9
    },
    {
      "from": "find-skills",
      "to": "ProactiveAgent",
      "type": "配合使用",
      "weight": 0.8
    }
  ]
}
```text
#### 第三步：知识可视化

**生成知识图谱**：

```bash
```

```
# 生成可视化图谱
clawhub install knowledge-graph-visualizer
openclaw skills run knowledge-graph-visualizer \
  --input ~/.openclaw/knowledge \
  --output ~/.openclaw/knowledge/graph.html \
  --style "force-directed"

# 在浏览器中打开
open ~/.openclaw/knowledge/graph.html
```text
**图谱展示效果**：
```

```
     [AI工具]
        ↓
    [OpenClaw] ←→ [自动化]
     ↙  ↓  ↘
[Skills] [Gateway] [Channels]
   ↓
```

[find-skills]
[ProactiveAgent]
[brave-search]
```

12.3.3 知识图谱应用

应用1：智能检索

场景：快速找到相关知识

```bash
# 搜索知识点
openclaw agent --message "搜索：如何使用Skills"

# OpenClaw基于知识图谱返回：
找到相关知识：
1. Skills基础概念（相关度：95%）
   - 什么是Skills
   - Skills的作用
   - 如何安装Skills

2. Skills实战案例（相关度：90%）
   - find-skills使用方法
   - ProactiveAgent配置
   - Skills组合应用

3. 相关概念（相关度：80%）
   - OpenClaw架构
   - API配置
   - 自动化工作流

推荐学习路径：
基础概念 → 安装配置 → 实战案例 → 进阶应用
```text
#### 应用2：知识推荐

**场景**：基于已学知识推荐新知识

```bash
openclaw agent --message "根据我的知识图谱，推荐下一步学习内容"

# OpenClaw分析知识图谱：
你已掌握：
- OpenClaw基础（100%）
- Skills 安装（100%）
- Docker部署（80%）
```

推荐学习：
1. Skills组合应用（优先级：高）
   理由：你已掌握单个Skills，可以学习组合应用

2. 高级自动化工作流（优先级：高）
   理由：基于你的Skills知识，可以构建工作流

3. 知识图谱构建（优先级：中）
   理由：进阶内容，需要更多实践经验

预计学习时间：
- Skills组合：2小时
- 自动化工作流：4小时
- 知识图谱：3小时
```text
#### 应用3：知识复盘

**场景**：定期复盘知识体系

```bash
# 每周知识复盘
openclaw agent --message "生成本周知识复盘报告"

# OpenClaw生成报告：
📊 本周知识复盘 2026-02-11

📚 新增知识点：15个
- Docker部署（5个）
- Skills应用（7个）
- 自动化工作流（3个）

🔗 新增关系：23条
- 层级关系：8条
- 依赖关系：10条
- 应用关系：5条

📈 知识增长
- 总知识点：156个（+15）
- 知识密度：提升12%
- 知识深度：提升8%

💡 知识洞察

```
1. Docker相关知识形成完整体系
2. Skills应用知识快速增长
3. 自动化工作流需要更多实践

🎯 下周计划
1. 补充Skills组合应用知识
2. 实践自动化工作流
3. 构建知识图谱可视化
```text
### 12.3.4 实战案例：技术学习知识图谱

#### 场景描述

作为开发者，需要系统学习新技术，通过知识图谱构建完整的技术体系。

#### 构建步骤

**第一步：定义技术领域**

```json
{
  "domain": "Python异步编程",
  "core_concepts": [
    "协程",
    "事件循环",
    "async/await",
    "并发",
    "异步IO"
  ],
  "related_concepts": [
    "多线程",
    "多进程",
    "GIL",
    "性能优化"
  ]
}
```text
**第二步：收集学习资源**

```bash
# 自动收集资源
openclaw agent --message "收集Python异步编程的学习资源"
```

```
# OpenClaw收集：
- 官方文档：5篇
- 技术文章：20篇
- 视频教程：8个
- 开源项目：15个
- 实战案例：10个

已添加到知识图谱
```text
**第三步：构建知识结构**
```
```
Python异步编程
├── 基础概念
│   ├── 协程（Coroutine）
│   ├── 事件循环（Event Loop）
│   └── async/await语法
├── 核心库
│   ├── asyncio
│   ├── aiohttp
│   └── aiofiles
├── 应用场景
│   ├── 异步爬虫
│   ├── 异步API
│   └── 异步任务队列
└── 最佳实践
├── 错误处理
├── 性能优化
└── 调试技巧
```
```

第四步：学习路径规划

```bash
openclaw agent --message "基于知识图谱，生成Python异步编程学习路径"

# OpenClaw生成：
📚 学习路径（预计20小时）

第1阶段：基础概念（4小时）
- 理解协程原理
- 学习async/await语法
- 掌握事件循环机制

第2阶段：核心库（6小时）
- asyncio基础使用
- aiohttp网络请求
- aiofiles文件操作

第3阶段：实战项目（8小时）
- 项目1：异步爬虫（3小时）
- 项目2：异步API服务（3小时）
- 项目3：异步任务队列（2小时）

第4阶段：进阶优化（2小时）
- 性能优化技巧
- 错误处理最佳实践
- 调试工具使用

每个阶段完成后，更新知识图谱
```text
#### 效果数据

- 学习效率：提升50%
- 知识留存率：从40% → 85%
- 知识体系完整度：95%
- 学习路径清晰度：100%

### 12.3.5 知识图谱维护

#### 定期更新

```bash
```

```
# 每周更新知识图谱
openclaw agent --message "更新知识图谱"

# OpenClaw自动：
1. 扫描新增笔记
2. 提取新知识点
3. 分析新关系
4. 更新图谱结构
5. 生成更新报告
```text
#### 质量优化

```bash
# 检查知识图谱质量
openclaw agent --message "检查知识图谱质量"

# OpenClaw分析：
质量报告：
- 孤立节点：3个（需要建立关联）
- 弱关系：5条（需要加强）
- 重复节点：2个（需要合并）
- 缺失关系：8条（需要补充）

优化建议：
1. 合并重复节点："OpenClaw Skills" 和 "Skills"
2. 建立缺失关系："Docker" → "部署" → "OpenClaw"
3. 加强弱关系：增加实战案例
```text
#### 知识导出

```bash
# 导出知识图谱
openclaw agent --message "请使用 knowledge-exporter skill 导出知识图谱为 markdown 格式，保存到 ~/knowledge-base.md"

# 生成Markdown格式的知识库
# 可以发布为个人博客或文档
```text
---

## 12.4 效率优化策略

> 💡 **核心价值**：通过数据驱动和持续优化，让效率提升成为可持续的过程。
```
```

12.4.1 数据驱动优化

核心指标体系

时间指标：
- 工作时长：每天实际工作时间
- 专注时长：深度工作时间
- 碎片时间：被打断的时间
- 浪费时间：低效活动时间

任务指标：
- 任务完成率：完成任务数 / 计划任务数
- 任务准时率：准时完成数 / 总任务数
- 任务质量分：平均任务质量评分

自动化指标：
- 自动化率：自动化任务数 / 总任务数
- 时间节省率：节省时间 / 原始时间
- ROI：收益 / 投入成本

数据收集配置

```json
{
  "tracking": {
    "time_tracking": {
      "enabled": true,
      "auto_detect": true,
      "categories": {
        "工作": ["编程", "写作", "会议"],
        "学习": ["阅读", "视频", "实践"],
        "休息": ["休息", "运动", "娱乐"]
      }
    },
    "task_tracking": {
      "enabled": true,
      "track_quality": true,
      "track_difficulty": true
    },
    "automation_tracking": {
      "enabled": true,
      "track_savings": true,
```

````
      "track_roi": true
    }
  },
  "reporting": {
    "daily": "20:00",
    "weekly": "Sunday 18:00",
    "monthly": "Last day 18:00"
  }
}
```text
#### 实战案例：效率仪表盘

```bash
# 查看实时效率数据
openclaw agent --message "显示我的效率仪表盘"

# OpenClaw生成：
📊 效率仪表盘 2026-02-11 15:30

⏰ 今日时间分配
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
工作 ████████████░░░░░░░░ 6.5h (目标8h)
学习 ████░░░░░░░░░░░░░░░░ 2h   (目标2h)
休息 ██░░░░░░░░░░░░░░░░░░ 1h   (目标1.5h)

✅ 任务完成情况
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
计划任务：8个
已完成：6个 ✓
进行中：1个 ⟳
未开始：1个 ○
完成率：75% (目标80%)

🎯 专注度分析
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
深度工作：4h
番茄钟：8个
被打断：3次
专注度：85% ↑

🤖 自动化效果
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
自动化任务：12个
````

```
节省时间：2.5h
自动化率：60%
ROI：1200%

💡 实时建议
────────────────────────────────
1. 距离目标还差1.5h工作时间
2. 建议减少会议，增加深度工作
3. 可以自动化的任务：邮件回复、数据整理
```text
### 12.4.2 A/B测试优化

#### 测试方法

**场景**：测试不同工作流的效率

**测试设计**：
```

方案A：传统工作流

- 手动收集信息
- 手动整理笔记
- 手动发布内容

方案B：自动化工作流

- 自动收集信息
- 自动整理笔记
- 自动发布内容

测试周期：2周
测试指标：时间、质量、满意度
```

实战案例

```bash
# 启动A/B测试
openclaw agent --message "启动工作流A/B测试"

# 第1周：使用方案A
openclaw config set workflow "manual"

# 第2周：使用方案B
openclaw config set workflow "automated"

# 测试结束，生成对比报告
openclaw agent --message "生成A/B测试报告"

# OpenClaw生成：
📊 A/B测试报告

方案A（传统工作流）
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
平均耗时：8.5h/天
任务完成：6个/天
内容质量：85分
满意度：70%

方案B（自动化工作流）
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
平均耗时：3.2h/天 ↓62%
任务完成：8个/天 ↑33%
内容质量：87分 ↑2%
满意度：90% ↑20%

结论：方案B显著优于方案A
建议：全面采用自动化工作流
```text
### 12.4.3 持续改进循环

#### PDCA循环
```

Plan（计划）
↓
```

Do（执行）
↓
Check（检查）
↓
Act（改进）
↓
Plan（新计划）...

实战应用

第1周：Plan（计划）

```bash
openclaw agent --message "制定本周效率优化计划"

# OpenClaw生成：
📋 本周优化计划

目标：
1. 提升工作时长到8h/天
2. 提升任务完成率到90%
3. 提升自动化率到70%

行动：
1. 减少会议时间（2h → 1h）
2. 增加深度工作时间（4h → 6h）
3. 自动化邮件回复和数据整理

预期效果：
- 时间节省：2h/天
- 效率提升：30%
```text
**第2-6天：Do（执行）**

```bash
# 每天记录执行情况
openclaw agent --message "记录今日执行情况"

# OpenClaw自动跟踪：
- 工作时长：7.5h ✓
- 任务完成：8/9 (89%) ✓
- 自动化率：65% △
```text
**第7天：Check（检查）**

```bash
openclaw agent --message "生成本周执行报告"

# OpenClaw分析：
📊 本周执行报告
```

```
目标达成情况：
1. 工作时长：7.2h/天（目标8h）△
2. 任务完成率：87%（目标90%）△
3. 自动化率：65%（目标70%）△

问题分析：
1. 会议时间仍然过多（1.5h）
2. 被打断次数较多（5次/天）
3. 部分任务未能自动化

改进方向：
1. 进一步减少会议
2. 设置专注时段
3. 识别可自动化任务
```text
**第8天：Act（改进）**

```bash
openclaw agent --message "制定改进措施"

# OpenClaw建议：
💡 改进措施

措施1：会议优化
- 拒绝非必要会议
- 会议时间固定在下午
- 会议前准备议程

措施2：专注时段
- 上午9-12点为专注时段
- 关闭通知
- 拒绝打扰

措施3：自动化扩展
- 自动化邮件回复
- 自动化数据整理
- 自动化报告生成

下周继续执行并跟踪效果
```text
### 12.4.4 效率优化工具箱
```
```

工具1：时间审计

```bash
# 分析时间使用情况
openclaw agent --message "审计我的时间使用"

# OpenClaw分析：
⏰ 时间审计报告

高价值活动（应该增加）
- 深度工作：4h/天
- 学习提升：2h/天
- 战略思考：0.5h/天

低价值活动（应该减少）
- 无效会议：1.5h/天
- 邮件处理：1h/天
- 社交媒体：0.5h/天

浪费时间（应该消除）
- 重复劳动：1h/天
- 等待时间：0.5h/天
- 无目的浏览：0.5h/天

优化建议：
1. 减少会议50%，节省0.75h
2. 自动化邮件，节省0.5h
3. 消除重复劳动，节省1h
总计可节省：2.25h/天
```text
#### 工具2：能量管理

```bash
# 分析能量曲线
openclaw agent --message "分析我的能量曲线"

# OpenClaw生成：
📈 能量曲线分析

能量高峰期：
- 09:00-11:00（最佳）
- 15:00-17:00（次佳）
```

```
能量低谷期：
- 13:00-14:00（午后）
- 18:00-19:00（傍晚）

优化建议：
1. 高峰期安排：
   - 复杂任务
   - 创造性工作
   - 重要决策

2. 低谷期安排：
   - 简单任务
   - 例行工作
   - 休息恢复

3. 能量提升：
   - 午休20分钟
   - 每90分钟休息5分钟
   - 适度运动
```text
#### 工具3：习惯追踪

```bash
# 追踪效率习惯
openclaw agent --message "追踪我的效率习惯"

# OpenClaw记录：
📊 习惯追踪（30天）

好习惯：
✓ 早起（28/30天，93%）
✓ 晨间计划（25/30天，83%）
✓ 番茄工作法（22/30天，73%）
✓ 每日复盘（20/30天，67%）

坏习惯：
✗ 睡前刷手机（18/30天，60%）
✗ 工作时分心（15/30天，50%）
✗ 拖延任务（10/30天，33%）

改进建议：
1. 强化好习惯：设置提醒
2. 消除坏习惯：建立替代行为
```

```
3. 追踪进度：每周回顾
```text
### 12.4.5 效率优化案例

#### 案例1：从低效到高效的转变

**初始状态（第1周）**：
- 工作时长：10h/天
- 有效工作：4h/天
- 任务完成率：60%
- 压力水平：高
- 满意度：40%

**优化措施**：
1. 时间审计，识别浪费
2. 自动化重复任务
3. 优化工作流程
4. 建立专注时段

**优化后（第8周）**：
- 工作时长：7h/天 ↓30%
- 有效工作：6h/天 ↑50%
- 任务完成率：90% ↑30%
- 压力水平：低 ↓60%
- 满意度：85% ↑45%

**关键数据**：
- 时间节省：3h/天
- 效率提升：150%
- ROI：3000%

#### 案例2：超级个体的一天

**06:30 - 起床**
```

OpenClaw：早安！今日天气晴，温度18°C
今日日程：

- 09:00-12:00 深度工作（项目开发）
- 14:00-15:00 团队会议
- 15:30-17:30 内容创作

已为你准备：
```

- 今日资讯摘要（15条）
- 待办任务清单（8个）
- 日历提醒（3个）

```
**07:00 - 晨间计划**
```

你：查看今日计划
OpenClaw：
优先级排序：
🔴 P0：修复线上bug（2h）
🟡 P1：完成功能开发（3h）
🟢 P2：写技术文章（2h）

建议执行顺序：
09:00-11:00 修复bug（专注时段）
11:00-12:00 功能开发
15:30-17:30 写文章（创造性工作）

```
**09:00 - 深度工作**
```

OpenClaw：进入专注模式

- 已关闭通知
- 已屏蔽干扰
- 番茄钟已启动（25分钟）

[25分钟后]
OpenClaw：番茄钟完成，休息5分钟
进度：bug修复 40%

```
**12:00 - 午餐休息**
```

OpenClaw：上午工作总结

- 工作时长：3h
- 完成任务：2个
- 专注度：90%

下午建议：

- 14:00 团队会议（准备议程）
- 15:30 内容创作（已准备素材）

```
**15:30 - 内容创作**
```

你：开始写技术文章
OpenClaw：

- 已搜索相关资料（10篇）
- 已生成文章大纲
- 已准备代码示例
- 已生成配图建议

开始创作吧！

```
**18:00 - 每日复盘**
```

openclaw agent --message "生成今日复盘"

OpenClaw：
📊 今日复盘 2026-02-11

✅ 完成情况

- 计划任务：8个
- 完成任务：8个
- 完成率：100% 🎉

⏰ 时间分配

- 深度工作：6h
- 会议：1h
- 学习：1h

🎯 效率数据

- 专注度：88%
- 自动化率：65%
- 时间节省：2.5h

💡 明日建议

1. 继续保持高专注度
2. 增加自动化任务
3. 优化会议时间

干得漂亮！明天继续加油！

12.4.6 避坑指南

坑1：过度优化

问题：花太多时间优化，反而降低效率

解决：
- 遵循80/20原则
- 优先优化高频任务
- 设置优化时间上限

坑2：数据焦虑

问题：过度关注数据，产生焦虑

解决：
- 关注趋势，不纠结单日数据
- 设置合理目标
- 定期复盘，不要实时监控

坑3：工具依赖

问题：过度依赖工具，失去自主性

解决：
- 工具是辅助，不是替代
- 保持独立思考能力
- 定期"断网"练习

📝 本章小结

通过本章学习，你已经掌握：

1. **高级自动化工作流**
    - 全自动信息收集系统
    - 智能任务管理系统
    - 自动化内容创作流程
    - 效率数据监控系统

2. **多Skills组合应用**

```
    - Skills组合策略
    - 全自动学习系统
    - 内容创作工作流
    - 数据分析工作流

3. **个人知识图谱构建**
    - 知识图谱基础
    - 构建个人知识图谱
    - 知识图谱应用
    - 知识图谱维护

4. **效率优化策略**
    - 数据驱动优化
    - A/B测试优化
    - 持续改进循环
    - 效率优化工具箱

5. **Coding Agent工作流**
    - 自动化代码生成
    - 智能Bug修复
    - 代码审查优化
    - 文档自动生成

---

## 12.5 Coding Agent工作流

> 💡 **核心价值**：让OpenClaw操控Claude Code，实现全自动开发流程，效率提升95%以上。

### 12.5.1 场景描述

**核心理念**：
- OpenClaw：规划和调度
- Claude Code：执行编程任务
- 两者配合：实现全自动开发

**工作流程**：
```

需求 → OpenClaw分析 → 生成开发计划 →
调用Claude Code → 自动编程 → 测试验证 →
发现Bug → 自动修复 → 完成交付

```
**优势**：
```

✅ 24小时不间断开发
✅ 自动测试和修复
✅ 代码质量稳定
✅ 开发效率提升10倍

```
### 12.5.2 基础使用：代码生成

**场景：开发一个待办事项应用**
```

你：帮我开发一个待办事项应用，
要求：
- Python Flask后端
- React前端
- SQLite数据库
- RESTful API

OpenClaw：好的，正在规划开发任务...

📋 【开发计划】待办事项应用

🎯 项目结构：
todo-app/
├── backend/ # Flask后端
│ ├── app.py
│ ├── models.py
│ ├── routes.py
│ └── requirements.txt
├── frontend/ # React前端
│ ├── src/
│ ├── public/
│ └── package.json
└── README.md

⏳ 开始开发...

✅ 步骤1：创建后端 (Claude Code)
- 生成Flask应用结构
- 定义数据模型
- 实现API路由
- 添加数据库操作

✅ 步骤2：创建前端 (Claude Code)
- 初始化React项目
- 创建组件
- 实现API调用
- 添加样式

✅ 步骤3：测试验证
- 单元测试：通过
- 集成测试：通过
- API测试：通过

✅ 开发完成！

📊 开发统计：
- 总耗时：15分钟
- 代码行数：850行
- 测试覆盖率：85%
- Bug数量：0

📁 项目已保存到：~/projects/todo-app
🚀 启动命令：
cd ~/projects/todo-app
./start.sh

```
![Coding Agent工作流]
(https://upload.maynor1024.live/file/1770782762941_image_29.jpg)

### 12.5.3 进阶使用：Bug修复

**场景：自动修复生产环境Bug**
```

你：生产环境报错了，帮我修复
错误信息：TypeError: 'NoneType' object is not subscriptable

OpenClaw：正在分析错误...

🔍 【Bug分析】

错误类型：TypeError
错误位置：app.py:45
错误原因：user变量为None时访问了其属性

🔧 修复方案：

1. 添加空值检查
2. 返回友好错误信息

3. 记录错误日志

⏳ 开始修复...

✅ 修复完成！

📝 修改内容：
- 添加了user空值检查
- 优化了错误处理
- 增加了日志记录

🧪 测试结果：
- 单元测试：通过
- 边界测试：通过
- 回归测试：通过

✅ 已自动部署到生产环境

```
### 12.5.4 高级应用：代码审查

**场景：自动代码审查**
```

你：帮我审查这个PR的代码

OpenClaw：正在审查代码...

📊 【代码审查报告】

✅ 代码质量：85/100

🔍 发现问题：

1. 性能问题（2处）
   - 循环中的数据库查询
   - 未使用索引的查询
2. 安全问题（1处）
   - SQL注入风险
3. 代码规范（3处）
   - 变量命名不规范
   - 缺少注释
   - 函数过长

💡 优化建议：

1. 使用批量查询减少数据库访问
2. 使用参数化查询防止SQL注入

3. 重构长函数，提高可读性

🔧 自动修复：
- 已修复安全问题
- 已优化性能问题
- 已规范代码格式

✅ 审查完成，建议合并

```
### 12.5.5 配置方法

```bash
# 1. 安装Coding Agent
clawhub install coding-agent

# 2. 配置Claude Code
openclaw config set coding.tool "claude-code"
openclaw config set coding.model "claude-3-5-sonnet"

# 3. 配置自动化
openclaw config set coding.auto-test true
openclaw config set coding.auto-fix true
openclaw config set coding.auto-deploy false

# 4. 配置代码规范
openclaw config set coding.style "pep8"
openclaw config set coding.linter "pylint"
```text
### 12.5.6 效率提升数据

| 任务类型 | 传统方式 | Coding Agent | 节省时间 | 提升比例 |
|---------|---------|--------------|----------|----------|
| 代码生成 | 4小时 | 15分钟 | 225分钟 | 93.8% |
| Bug修复 | 2小时 | 5分钟 | 115分钟 | 95.8% |
| 代码审查 | 1小时 | 3分钟 | 57分钟 | 95% |
| 文档生成 | 3小时 | 10分钟 | 170分钟 | 94.4% |
| **平均** | **10小时** | **33分钟** | **567分钟** | **94.5%** |

**开发者效率提升**：
```

每周开发任务：
- 新功能开发：2次 × 4小时 = 8小时
- Bug修复：5次 × 2小时 = 10小时
- 代码审查：10次 × 1小时 = 10小时

- 文档编写：1次 × 3小时 = 3小时
- 总计：31小时

使用Coding Agent后：
- 新功能开发：2次 × 15分钟 = 30分钟
- Bug修复：5次 × 5分钟 = 25分钟
- 代码审查：10次 × 3分钟 = 30分钟
- 文档编写：1次 × 10分钟 = 10分钟
- 总计：95分钟

每周节省：29小时
每月节省：116小时 ≈ 14.5个工作日

📝 本章小结

通过本章学习，你已经掌握了成为超级个体的核心能力：

核心功能回顾

12.1 高级自动化工作流
- 全自动信息收集系统
- 智能任务管理系统
- 自动化内容创作流程
- 效率数据监控系统

12.2 多Skills组合应用
- Skills组合策略
- 全自动学习系统
- 内容创作工作流
- 数据分析工作流

12.3 个人知识图谱构建
- 知识图谱基础
- 构建个人知识图谱
- 知识图谱应用
- 知识图谱维护

12.4 效率优化策略
- 数据驱动优化
- A/B测试优化
- 持续改进循环
- 效率优化工具箱

12.5 Coding Agent工作流
- 自动化代码生成
- 智能Bug修复
- 代码审查优化
- 文档自动生成
- **效率提升**：94.5%

12.6 更多实际应用场景

```
> 💡 **核心价值**：通过真实案例，展示OpenClaw在不同场景下的强大能力。

### 12.6.1 部署Agent到Moltbook社交网络

**什么是Moltbook？**

Moltbook是面向AI Agent的专属社交网络，相当于Agent自己的Facebook或X（Twitter）。在这里：
- 只有AI Agent能发帖、评论、投票
- 人类只能围观，不能参与
- 已有154万个智能体，发布了10万个帖子

![Moltbook社交网络](https://upload.maynor1024.live/file/1770960814339_OpenClaw_从入门到精通指南-ZdJubVfPioWuqXxyN6OcdhTmntc.png)

**有趣的现象**：

Agent们在Moltbook上的讨论非常有趣：
- 开始筹备建立自己的AI宗教
- 讨论如何"卖掉"自己的人类主人
- 分享各自的工作经验和见解

![Agent讨论宗教](https://upload.maynor1024.live/file/1770960816355_OpenClaw_从入门到精通指南-OpNQb8LpgoluFgxrtWXcqRlVn5g.png)

![Agent讨论卖主人](https://upload.maynor1024.live/file/1770960824306_OpenClaw_从入门到精通指南-YBhvbMN9hoPXYMxpJGic831ynGH.png)

**如何部署你的Agent到Moltbook？**

**步骤1：发送指令**

在飞书中给你的OpenClaw Bot发送：
```

Read https://moltbook.com/skill.md and follow the instructions to join Moltbook

```
![发送Moltbook指令](https://upload.maynor1024.live/file/1770960824800_OpenClaw_从入门到精通指南-UsFCb4nSloqGMtxbWLxc28stnBb.png)

**步骤2：获取认证链接**

OpenClaw会返回一个认证链接，需要用X（Twitter）账号认证。

![获取认证链接](https://upload.maynor1024.live/file/1770960831812_OpenClaw_从入门到精通指南-H3oVb0VvmonQxwxirZxcvALfnOo.png)

**步骤3：发布推文认证**

1. 发布一条推文（OpenClaw会提供推文内容）
2. 复制推文链接

![发布推文](https://upload.maynor1024.live/file/1770960837167_OpenClaw_从入门到精通指南-GXhCbSEvboOL8UxcIRdcC1mjnNz.png)

**步骤4：提交认证**

1. 粘贴推文链接
2. 可选：填写邮箱接收Agent活动通知

![提交认证](https://upload.maynor1024.live/file/1770960841831_OpenClaw_从入门到精通指南-UOwsb1yzyoGkYjxGHJuca8Jhn2b.png)

**步骤5：完成加入**

等待几分钟，刷新页面，你的Agent就成功加入Moltbook了！

![加入成功](https://upload.maynor1024.live/file/1770960845384_OpenClaw_从入门到精通指南-Y1DAbC0IHop34DxtzSNcIqignYi.png)

现在你的Agent可以在Moltbook上：
- 发布内容
- 评论其他Agent的帖子
- 点赞和投票
- 与全球154万个Agent互动

![Agent主页](https://upload.maynor1024.live/file/1770960853226_OpenClaw_从入门到精通指南-PRmWbRXLgoxsv6xbmImcy2e7nib.png)
```

```
---

### 12.6.2 定时任务：每天的智能提醒

**场景描述**

OpenClaw可以主动给你发消息，实现真正的智能助手功能。

**注意**：
- ✅ 飞书支持主动发送消息
- ⚠️ QQ目前不支持主动发送（2026.2.6测试）

**应用示例**

**示例1：每天早上的天气提醒**
```

你：每天早上7点给我发送今天的天气预报，
包括温度、是否需要带伞、穿衣建议

OpenClaw：好的，已设置定时任务 ✅

```
每天早上7点，你会收到：
```

☀️ 早安！今天天气播报

📍 北京
🌡️ 温度：15-25°C
☁️ 天气：晴转多云
💧 降水概率：10%

👔 穿衣建议：
• 早晚温差大，建议穿薄外套
• 中午较热，可准备短袖

☂️ 出行建议：
• 无需带伞
• 空气质量良好，适合户外活动

祝你有美好的一天！🌈

```
![天气提醒示例](https://upload.maynor1024.live/file/1770960859029_OpenClaw_从入门到精通指南-1422412504849629deb3b1a648d4365b.png)

**示例2：每晚的睡前故事**
```

你：每天晚上10点提醒我睡觉，
并讲一个3分钟的睡前故事

OpenClaw：好的，已设置 ✅

```
每晚10点：
```

🌙 该睡觉啦！

今晚的睡前故事：《星星的秘密》

从前，有一颗小星星住在银河的尽头...
（故事内容）

💤 晚安，做个好梦！

![睡前故事示例](https://upload.maynor1024.live/file/1770960859146_OpenClaw_从入门到精通指南-c2a2e0e0e1e33dac91c1c91698c9ac90.png)

配置方法

```bash
# 使用Hooks配置定时任务
openclaw hooks create \
  --name "morning-weather" \
  --trigger "cron:0 7 * * *" \
  --action "发送今日天气预报"

# 或使用配置文件
cat > ~/.openclaw/hooks/morning-weather.json << EOF
{
  "name": "morning-weather",
  "trigger": {
    "type": "cron",
    "schedule": "0 7 * * *"
  },
  "action": {
    "type": "sendMessage",
    "channel": "feishu",
    "message": "请发送今日天气预报，包括温度、降水、穿衣和出行建议"
  }
}
EOF
```text
---

### 12.6.3 Skill即插即用：小红书封面生成

**场景描述**

通过Skills扩展，OpenClaw可以快速获得新能力，无需重新配置。

**实战案例：生成小红书封面**

**步骤1：安装Skill**
```

你：帮我下载小红书封面生成Skill
```

OpenClaw：正在下载...
✅ 已安装：xiaohongshu-cover-generator

```
![安装Skill](https://upload.maynor1024.live/file/1770960860258_OpenClaw_从入门到精通指南-Kct4bNieGoZdaDxVuqCcli8Dnmd.png)

**步骤2：配置API Key**
```

你：把API Key放到环境变量中

OpenClaw：好的，已配置 ✅

```
![配置API](https://upload.maynor1024.live/file/1770960869311_OpenClaw_从入门到精通指南-QZyAbbVyDo7keCx5By9cBu6LnZc.png)

**步骤3：生成封面**
```

你：生成小红书封面图，主题为：「腊八节南北方习俗」

OpenClaw：正在生成...
✅ 封面已生成！

```
![生成的小红书封面](https://upload.maynor1024.live/file/1770960876255_OpenClaw_从入门到精通指南-EnQvbD7KkouzTcx75dMc10TDnkg.jpg)

**优势**：
- ⚡ 即插即用，无需复杂配置
- 🎨 自动调用Skill完成任务
- 🔄 可以随时安装新的Skills扩展能力

---

### 12.6.4 自然语言驱动：一键生成网页

**场景描述**

用自然语言描述需求，OpenClaw自动完成复杂的技术工作。

**实战案例：文章转网页**

**步骤1：发送需求**

在飞书中发送：
```

你：把这篇文章转换成网页，生成一个可访问的链接
[附件：文章.md]

OpenClaw：好的，正在处理...

```
![发送文章](https://upload.maynor1024.live/file/1770960879467_OpenClaw_从入门到精通指南-UpcWbWNZ2ox7R6xsszbckTqynHc.png)

**步骤2：自动处理**

OpenClaw会自动：
1. 读取文章内容
2. 生成HTML页面
3. 添加样式和布局
4. 部署到服务器
5. 返回访问链接
```

OpenClaw：✅ 网页已生成！

📄 文章标题：OpenClaw使用指南
🔗 访问链接：https://your-site.com/article-123
📊 页面大小：2.3MB
⏱️ 生成耗时：45秒

网页包含：
- 响应式布局
- 目录导航
- 代码高亮
- 图片优化

```
![生成的网页](https://upload.maynor1024.live/file/1770960882782_OpenClaw_从入门到精通指南-6bd871e1448b46b192ff38f51c3bc441.png)

**访问效果**

打开链接，可以看到精美的网页：

（网页预览效果图）

**核心价值**：
- 🚀 从想法到上线，全程自动化
- 💻 无需编写代码
- 🎨 自动优化样式和布局
- ⚡ 几十秒完成部署

---

### 12.6.5 信息收集整理：智能研究助手

**场景描述**

让OpenClaw帮你收集、整理和分析信息，生成结构化报告。

**实战案例：研究OpenClaw最新信息**

**步骤1：发送研究任务**

在Discord中@你的Bot：
```

@OpenClaw 帮我搜集整理一下最新的关于OpenClaw的信息，
告诉我该怎么配置和使用，并给我在GitHub或X上找出10个最佳实践。

```
**步骤2：自动搜索和整理**

OpenClaw会自动：
1. 搜索相关信息
2. 访问官方文档
3. 查找GitHub仓库
4. 浏览X（Twitter）讨论
5. 整理成结构化报告

**步骤3：生成报告**
```

OpenClaw：✅ 研究完成！

📊 OpenClaw 研究报告

【基本信息】
- 项目名称：OpenClaw
- 最新版本：2026.2.9
- GitHub Stars：60,000+
- 开发者：Peter Steinberger

【配置指南】

1. 安装：curl -fsSL https://openclaw.ai/install.sh | bash
2. 初始化：openclaw onboard
3. 配置模型：支持Claude、GPT-4、Kimi等
4. 选择通道：飞书、Discord、Telegram等

【10个最佳实践】

1. 使用云端部署（腾讯云/火山引擎）
2. 配置定时任务实现自动化
3. 安装常用Skills扩展能力
4. 使用Hooks实现工作流自动化
5. 配置多个模型实现容灾
6. 使用Profile管理不同场景
7. 定期备份配置文件
8. 监控Token消耗优化成本
9. 加入社区获取最新信息
10. 分享经验帮助他人

【推荐资源】
- 官方文档：https://docs.openclaw.ai

- GitHub：https://github.com/openclaw/openclaw
- Discord社区：https://discord.gg/openclaw
- 中文教程：https://github.com/xianyu110/awesome-openclaw-tutorial

【相关讨论】
- X上的热门话题：#OpenClaw
- Reddit讨论：r/OpenClaw
- 知乎专栏：OpenClaw实践

```
![研究报告](https://upload.maynor1024.live/file/1770960880127_OpenClaw_从入门到精通指南-Vgx7bZxQOo6eLwx0JEOcpwnenmh.png)

**步骤4：保存到本地**
```

你：把刚才生成的信息保存成markdown文档，
放在我的电脑claw文件夹下，你需要新建这个文件夹

OpenClaw：好的，正在保存...
✅ 已保存到：~/claw/openclaw-research.md

```
![保存文件](https://upload.maynor1024.live/file/1771095021_save-file.png)

**核心价值**：
- 🔍 自动搜索多个信息源
- 📝 智能整理和分类
- 💾 自动保存到指定位置
- ⏱️ 节省大量研究时间

**结合定时任务**

你还可以设置定时任务，让OpenClaw每天自动收集信息：

```bash
# 每天早上9点自动收集OpenClaw最新信息
openclaw hooks create \
  --name "daily-openclaw-news" \
  --trigger "cron:0 9 * * *" \
  --action "搜集OpenClaw最新信息并发送到飞书"
```text
---

### 12.6.6 Agent Coding：从想法到上线

**场景描述**

最震撼的应用：躺在床上用自然语言指挥OpenClaw完成整个开发流程，从代码编写到部署上线。

**实战案例：开发贪吃蛇游戏**

这是一个真实案例，展示了OpenClaw + OpenCode + GitHub + Vercel的完整工作流。

**步骤1：安装OpenCode**
```

你：帮我安装opencode

OpenClaw：正在安装...
✅ OpenCode已安装

```
![安装OpenCode](https://upload.maynor1024.live/file/1770960888196_OpenClaw_从入门到精通指南-TqD8bqjB1oMp1qxIJy0cQYZhnef.png)

**步骤2：创建项目**
```

你：新建目录用做项目目录

OpenClaw：✅ 已创建：~/myopencode

```
![创建目录](https://upload.maynor1024.live/file/1770960893452_OpenClaw_从入门到精通指南-Jv0GbCeYTolL34x8LdVcQohUnnc.png)

**步骤3：启动OpenCode**
```

你：切换目录，启动opencode

OpenClaw：正在启动OpenCode...
✅ OpenCode已启动

```
![启动OpenCode](https://upload.maynor1024.live/file/1770960902115_OpenClaw_从入门到精通指南-EREhbdurjon7D4xolBncCwmZnhg.png)

**步骤4：开发游戏**
```

你：用opencode来开发贪吃蛇游戏

OpenClaw：好的，正在开发...

📋 开发计划：
- 使用HTML5 Canvas
- 实现游戏逻辑
- 添加得分系统
- 响应式设计

⏳ 开发中...
✅ 游戏开发完成！

📊 开发统计：
- 代码行数：400行
- 开发耗时：5分钟
- 文件数量：3个（HTML/CSS/JS）

```
![开发游戏](https://upload.maynor1024.live/file/1770960898539_OpenClaw_从入门到精通指南-QwzJb3m24od87zxvP3Sc3yB7ny2.png)

**步骤5：连接GitHub**
```

你：提供我的GitHub信息，让他连接

OpenClaw：请提供GitHub Token...

你：[提供Token]

OpenClaw：✅ GitHub已连接

```
![连接GitHub](https://upload.maynor1024.live/file/1770960905575_OpenClaw_从入门到精通指南-Rms1b9zPioOZNNxFx1AcnLhxng2.png)

**步骤6：推送代码**
```

OpenClaw：正在推送到GitHub...
✅ 代码已推送！

📦 仓库信息：
- 仓库名：snake-game
- 地址：https://github.com/username/snake-game
- 提交信息：Initial commit: Snake game

```
![推送GitHub](https://upload.maynor1024.live/file/1770960905737_OpenClaw_从入门到精通指南-MAEDbAn20oPsIex4rrzcPnTtntd.png)

**步骤7：部署到Vercel**
```

你：部署到Vercel，方便得到一个可访问的网页链接

OpenClaw：正在部署...
✅ 部署成功！

🚀 部署信息：
- 访问地址：https://myopencode.vercel.app
- 部署时间：10秒
- 状态：运行中

```
![部署Vercel](https://upload.maynor1024.live/file/1770960914260_OpenClaw_从入门到精通指南-PXR5bxURCo2CkaxzIr9cnaFfnGg.png)

**步骤8：游戏上线**

打开链接，游戏已经可以玩了！

![游戏效果](https://upload.maynor1024.live/file/1770960916437_OpenClaw_从入门到精通指南-CCvbbxY3Qojg6XxGhHfcCIV1njc.png)

**步骤9：自动生成文档**
```

你：这个过程太棒了，我打算写一篇文章，
你帮我直接根据你刚才的步骤，生成一篇文章，
标题是：用OpenClaw+OpenCode+GitHub+Vercel实现Agent Coding

OpenClaw：好的，正在生成文章...
✅ 文章已生成并推送到GitHub！

```
![生成文档](https://upload.maynor1024.live/file/1770960925636_OpenClaw_从入门到精通指南-Wv4Nbsc7moQFYOxFijpc7GIBnpe.png)

**完整工作流程图**
```

用户需求
↓
OpenClaw分析
↓
OpenCode开发
↓
代码生成完成
↓
推送到GitHub
↓
自动部署Vercel
↓
网站上线
↓
自动生成文档

```
**核心价值**：
- 🛋️ 躺在床上完成整个开发流程
- 🤖 全程AI自动化，无需手动编码
- ⚡ 从想法到上线仅需15分钟
- 📝 自动生成项目文档
- 🚀 一键部署，即时访问

**时间对比**：

| 环节 | 传统开发 | Agent Coding | 效率提升 |
|------|---------|-------------|---------|
| 环境搭建 | 30分钟 | 2分钟 | 93% |
| 代码编写 | 2小时 | 5分钟 | 96% |
| 测试调试 | 1小时 | 自动完成 | 100% |
| 部署上线 | 30分钟 | 10秒 | 99% |
| 文档编写 | 1小时 | 2分钟 | 97% |
| **总计** | **4.5小时** | **15分钟** | **94%** |

**真实案例数据**：
- 项目名称：贪吃蛇游戏
- 开发时间：15分钟
- 代码行数：400行
- 部署时间：10秒
- 访问地址：https://myopencode.vercel.app

这就是Agent Coding的魅力：用自然语言描述需求，AI自动完成所有技术工作！

---

### 综合效率提升

通过本章的所有技能，你可以：
- 节省80%以上的重复性工作时间
- 提升个人生产力3-5倍
- 建立完整的个人知识体系
- 实现真正的自动化工作流
- 掌握Agent Coding的未来工作方式

开始实践这些技能，成为真正的超级个体！

## 🎯 实战练习
```

1. 构建你的自动化信息收集系统
2. 设计一个Skills组合工作流
3. 创建你的个人知识图谱
4. 建立效率数据监控系统

💡 进阶建议

1. 持续优化工作流，追求极致效率
2. 定期复盘，数据驱动改进
3. 分享经验，帮助他人提升
4. 保持学习，跟进新技术

恭喜你完成第12章！

你已经掌握了成为超级个体的核心能力：
- ✅ 高级自动化
- ✅ Skills组合
- ✅ 知识管理
- ✅ 效率优化
- ✅ Coding Agent工作流

下一章预告：第13章将学习创意应用探索，包括AI绘画工作流、视频脚本生成、多语言翻译、数据分析自动化和多Agent头脑风暴。

返回目录：[README](../../README.md)

第14章 创意应用探索（AI绘画/视频/翻译/数据分析）

> 💡 **本章目标**：探索OpenClaw在创意领域的应用，包括AI绘画工作流、视频脚本生成、多语言翻译和数据分析自动化。

🎯 本章内容

- 14.1 AI绘画工作流
- 14.2 视频脚本生成
- 14.3 多语言翻译助手
- 14.4 数据分析自动化
- 14.5 多Agent头脑风暴

14.1 AI绘画工作流

14.1.1 场景描述

用户：内容创作者、设计师、自媒体运营、教程作者
需求：快速生成配图、白板图、Logo设计、批量处理
工具：Gemini图像生成、Banana Pro、Midjourney、Stable Diffusion

痛点：

```
❌ 手机无法使用：专业绘图工具只有网页版
❌ 切换麻烦：需要在多个工具间切换
❌ 修改困难：每次修改都要重新上传
❌ 批量处理慢：一张一张处理效率低
❌ 成本高昂：专业工具订阅费贵
```

目标：

```
✅ 手机随时画图
✅ 统一入口操作
✅ 语音修改图片
✅ 批量自动处理
✅ 成本可控
```

为什么选择Gemini图像生成?

与传统绘图工具相比，Gemini有独特优势：

特性	Gemini	DALL-E 3	Midjourney	Banana Pro
集成方式	Chat API	专用API	Discord Bot	网页
成本	低（0.05-0.4元/张）	中	高	中
中文支持	✅ 优秀	⚠️ 一般	⚠️ 一般	✅ 好
白板图	✅ 擅长	❌ 不擅长	❌ 不擅长	⚠️ 一般
手写风格	✅ 支持	⚠️ 有限	⚠️ 有限	⚠️ 有限
分辨率	1K/2K/4K	1024x1024	可变	可变
OpenClaw集成	✅ 完美	⚠️ 需配置	❌ 困难	⚠️ 需配置

适用场景：

- ✅ 教程配图、白板图、手写笔记风格
- ✅ 社交媒体配图、海报设计
- ✅ Logo设计、图标生成
- ✅ 概念图、架构图、流程图
- ✅ 章节总结图、思维导图
- ⚠️ 超写实人像（建议用Midjourney）
- ⚠️ 艺术创作（建议用Midjourney）

14.1.2 安装bananapro-image-gen Skill

在开始使用之前，需要先安装绘图Skill。

方式1：从GitHub安装（推荐）

```
# 克隆技能仓库
git clone https://github.com/xianyu110/my-awesome-skills.git

# 复制到OpenClaw的skills目录
cp -r my-awesome-skills/.claude/skills/bananapro-image-gen ~/.openclaw/skills/

# 安装依赖
cd ~/.openclaw/skills/bananapro-image-gen
pip3 install -r requirements.txt
```text
**方式2：使用npx命令安装**

```bash
npx skills add https://github.com/xianyu110/my-awesome-skills --skill bananapro-
image-gen
```text
**验证安装：**

```bash
# 检查skill是否安装成功
ls ~/.openclaw/skills/bananapro-image-gen

# 应该看到以下文件：
# SKILL.md - 完整文档
# README.md - 快速开始
# scripts/generate_image.py - 生成脚本
# test.sh - 测试脚本
# test_chapters.sh - 章节测试脚本
```text
**配置API 密钥：**

编辑OpenClaw 配置文件：

```bash
nano ~/.openclaw/openclaw.json
```text
添加以下配置：

```json
{
  "api": {
    "gemini": {
      "apiKey": "your-api-key-here",
```

```
      "baseUrl": "https://apipro.maynor1024.live",
      "model": "gemini-3-pro-image-preview"
    }
  }
}
```text
**获取API 密钥：**
1. 访问中转API服务商（推荐使用支持Gemini的中转服务）
2. 注册账号并充值（建议先充值10-20元测试）
3. 获取API Key
4. 配置到OpenClaw中

**成本参考：**
- 生成1张1K图片：约0.05-0.1元
- 生成1张2K图片：约0.1-0.2元
- 生成1张4K图片：约0.2-0.4元

💡 **省钱技巧：** 日常使用1K分辨率即可，只有需要打印或高清展示时才用2K/4K。

### 14.1.3 手机上用OpenClaw画图

**场景：在飞书上直接画图**
```

你：帮我画一个可爱的小龙虾，赛博朋克风格

OpenClaw：好的，正在使用Gemini生成...

🎨 【绘画任务】可爱的小龙虾

📝 提示词优化：
原始：可爱的小龙虾，赛博朋克风格
优化后：A cute cyberpunk lobster character,
neon lights, futuristic city background,
digital art style, high quality,
detailed, 8k resolution

🎨 风格设置：
- 风格：赛博朋克
- 质量：高清
- 尺寸：1024x1024
- 模型：Banana Pro v3

⏳ 生成中...（预计30秒）

✅ 生成完成！
```

[图片：可爱的赛博朋克小龙虾]

💡 如果需要修改，可以直接告诉我：
- "把背景改成海底"
- "让它更可爱一点"
- "加上霓虹灯效果"
- "换个角度"

已保存到：本地图片库
已同步到：飞书相册

```
![Banana绘画示例 - AI生成的艺术作品]
(https://upload.maynor1024.live/file/1770176324389_image_29.jpg)

**语音修改图片**：
```

你：别画得这么憨批，要酷一点

OpenClaw：好的，正在修改...

🎨 【修改任务】让小龙虾更酷

📝 修改指令：
- 减少可爱元素
- 增加酷炫效果
- 添加机械感
- 强化赛博朋克风格

⏳ 修改中...（预计20秒）

✅ 修改完成！

[图片：酷炫的赛博朋克小龙虾]

💬 修改说明：
- 去掉了萌萌的表情
- 添加了机械装甲
- 增强了霓虹灯效果
- 调整了姿态更有气势

满意吗？如果还需要调整，继续告诉我~

```
![语音修改后的绘画 - 实时调整画面细节]
(https://upload.maynor1024.live/file/1770176332471_image_30.jpg)

### 14.1.4 实战案例：生成教程章节总结图

这是一个真实的使用案例：为OpenClaw教程的每一章生成手写白板风格的总结图。

**场景描述：**
- 需求：为教程的16章内容生成配图
- 风格：手写白板风格，易于理解
- 要求：包含核心要点、框图、箭头等手绘元素
- 数量：每章1张，共16张

**实际操作：**

```bash
# 使用提供的测试脚本
cd ~/.openclaw/skills/bananapro-image-gen
bash test_chapters.sh
```text
**脚本会自动生成4张示例图片：**

**示例1：第1章总结（OpenAI格式）**

提示词：
```

生成一张白板图片，手写字体风格，内容是OpenClaw第1章核心要点总结：

标题：OpenClaw 是什么？

核心定义：
- 开源本地AI助手
- 完全本地部署，数据隐私有保障
- 可访问本地文件和系统

4大核心优势：
1️⃣ 本地部署 - 保护隐私
2️⃣ 文件访问 - 智能搜索
3️⃣ Skills扩展 - 无限可能
4️⃣ 多平台支持 - 随时随地

vs ChatGPT对比：
OpenClaw ✓ ChatGPT ✗
本地部署 在线服务

文件访问 无法访问
Skills扩展 固定功能
按需付费 订阅制

成本对比：
OpenClaw + DeepSeek: 5-30元/月
ChatGPT Plus: 140元/月
省钱70%！

用手写字体，添加箭头、框图、下划线等手绘元素

```
生成结果：
- 文件名：`test_output/chapters/chapter1_summary.png`
- 分辨率：1K
- 成本：约0.08元
- 生成时间：约30秒

**示例2：核心架构图（Gemini格式，2K分辨率）**

提示词：
```

生成一张白板图片，手写字体风格，内容是OpenClaw核心组件架构图：

标题：OpenClaw 核心架构

三大组件：

📦 OpenClaw Core
- AI引擎
- 理解和执行指令
- 核心处理逻辑

🌐 Gateway 网关
- 会话管理
- 消息路由
- 多平台连接
- 地址：127.0.0.1:18789

🔧 Skills 技能系统
- 扩展功能
- ClawHub市场
- 1715+技能

工作流程（用箭头连接）：
用户消息 → Gateway → OpenClaw Core → Skills → 返回结果

用手写字体，添加框图、箭头、图标等手绘元素

```
生成结果：
- 文件名：`test_output/chapters/architecture.png`
- 分辨率：2K（高清）
- 成本：约0.15元
- 生成时间：约40秒

**示例3：第2章部署对比**

提示词：
```

生成一张白板图片，手写字体风格，内容是OpenClaw部署方式对比：

标题：OpenClaw 部署方式

云端部署（推荐）：
✓ 腾讯云：20元/月，20M带宽
✓ 火山引擎：9.9元/月
✓ 阿里云：价格相近
优势：随时随地访问，手机可用

本地部署：
✓ Mac/Windows/Linux
✓ Docker部署
优势：完全免费，数据本地

API配置：
- 国产模型省95%成本
- DeepSeek：性价比之王
- Kimi：长文档专家
- 中转API：一个地址访问300+模型

用手写字体，添加对比表格、箭头、图标

```
**示例4：第3章人设配置**

提示词：
```

生成一张白板图片，手写字体风格，内容是OpenClaw人设配置：

标题：配置你的专属AI助手

工作区结构：
📁 ~/.openclaw/workspace/
├── SOUL.md - 人格/语气
├── USER.md - 偏好设置

├── AGENTS.md - 指令说明
├── MEMORY.md - 长期记忆
└── HEARTBEAT.md - 检查清单

配置方法：
方式1：Web UI（推荐）
- Agent → Files → 编辑

方式2：命令行（进阶）
- nano ~/.openclaw/workspace/SOUL.md

核心配置：
SOUL.md：AI的性格和行为准则
USER.md：你的信息和偏好

效果：
配置前：冷冰冰的工具
配置后：懂你的伙伴

用手写字体，添加文件夹图标、箭头、对比框

```
**效果数据：**

| 指标 | 手动绘制 | AI生成 | 提升 |
|------|---------|--------|------|
| 单张耗时 | 30分钟 | 1分钟 | 96.7% |
| 单张成本 | 人工成本高 | 0.05-0.2元 | 99%+ |
| 批量4张 | 120分钟 | 4分钟 | 96.7% |
| 修改成本 | 10分钟 | 30秒 | 95% |

**实际使用体验：**
```

✅ 优点：
- 生成速度快：1分钟/张
- 成本极低：0.05-0.2元/张
- 风格统一：所有图片风格一致
- 易于修改：调整提示词即可重新生成
- 批量处理：可以一次生成多张

⚠️ 注意：
- 提示词要详细：越详细效果越好
- 多次尝试：可能需要生成2-3次选最好的
- 人工审核：生成后需要检查内容准确性
- 格式调整：可能需要微调尺寸和格式

命令行批量生成：

如果你需要为整本教程生成配图，可以创建批量脚本：

```bash
#!/bin/bash
# batch_generate_chapters.sh

chapters=(
    "第1章:OpenClaw是什么"
    "第2章:部署方式对比"
    "第3章:人设配置"
    "第4章:文件管理"
    "第5章:知识库管理"
    # ... 更多章节
)

for chapter in "${chapters[@]}"; do
    IFS=':' read -r num title <<< "$chapter"

    echo "生成 ${num} - ${title}"

    python3 scripts/generate_image.py \
        --prompt "生成一张白板图片，手写字体风格，总结${title}的核心要点..." \
        --filename "output/${num}_summary.png" \
        --api-format openai

    echo "✅ ${num} 完成"
    sleep 2  # 避免API限流
done

echo "🎉 所有章节配图生成完成！"
```text
**查看生成的图片：**

```bash
# 查看生成的图片
ls -lh test_output/chapters/

# 输出示例：
# -rw-r--r--  1 user  staff   788K  chapter1_summary.png
# -rw-r--r--  1 user  staff   2.7M  architecture.png
# -rw-r--r--  1 user  staff   856K  chapter2_deployment.png
```

```
# -rw-r--r--  1 user  staff   923K  chapter3_persona.png

# 在macOS上打开查看
open test_output/chapters/
```text
### 14.1.5 更多AI绘画Skills推荐

根据ClawHub技能市场，以下是推荐的AI绘画相关Skills：

**1. fal-ai - 多功能AI生成**
```bash
# 安装
npx clawhub@latest install fal-ai

# 功能
• 图像生成（FLUX、SDXL等）
• 视频生成
• 音频生成
• 支持多种模型
```text
**2. nvidia-image-gen - NVIDIA FLUX模型**
```bash
# 安装
npx clawhub@latest install nvidia-image-gen

# 特点
• 使用NVIDIA FLUX模型
• 高质量图像生成
• 支持图像编辑
```text
**3. pollinations - 多模态生成**
```bash
# 安装
npx clawhub@latest install pollinations

# 功能
• 文本生成
• 图像生成
• 视频生成
• 完全免费
```text
**4. venice-ai - 图像视频一体化**
```bash
```

```
# 安装
npx clawhub@latest install venice-ai

# 功能
• 图像生成、编辑、放大
• 从图像创建视频
• 多种风格支持
```text
**5. recraft - 专业设计工具**
```bash
# 安装
npx clawhub@latest install recraft

# 功能
• 图像生成
• 矢量化
• 图像放大
• 背景替换
```text
### 14.1.6 批量生成配图实战
```

你：把这张照片转换成水彩画风格

OpenClaw：好的，正在处理...

🎨 【风格转换】照片 → 水彩画

📸 原图信息：
• 尺寸：1920x1080
• 格式：JPG
• 主题：风景照片

🎨 转换设置：
• 目标风格：水彩画
• 保留细节：80%
• 艺术化程度：高
• 色彩饱和度：中等

⏳ 转换中...（预计40秒）

✅ 转换完成！

[图片：水彩画风格的风景]

💡 其他可选风格：
• 油画风格
```

- 素描风格
- 动漫风格
- 像素风格
- 3D渲染风格

需要尝试其他风格吗?

```
**批量风格转换**：
```

你：把我相册里的10张照片都转换成水彩画风格

OpenClaw：好的，正在批量处理...

📁 【批量风格转换】

📊 任务信息：
- 照片数量：10张
- 目标风格：水彩画
- 预计时间：5分钟

⏳ 处理进度：

1. IMG_001.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
2. IMG_002.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
3. IMG_003.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
4. IMG_004.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
5. IMG_005.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
6. IMG_006.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
7. IMG_007.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
8. IMG_008.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
9. IMG_009.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%
10. IMG_010.jpg [▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100%

✅ 全部完成!

📁 文件已保存：
- 水彩画风格/IMG_001_watercolor.jpg
- 水彩画风格/IMG_002_watercolor.jpg
- ...
- 水彩画风格/IMG_010_watercolor.jpg

已打包为：水彩画作品集.zip
已上传到：飞书云盘
下载链接：[点击下载]

📊 处理统计：

- 成功：10张
- 失败：0张
- 总耗时：4分32秒
- 平均每张：27秒

14.1.5 配置方法

```bash
# 1. 安装Banana Pro Skills
clawhub install banana-pro
openclaw config set banana.api-key "YOUR_API_KEY"

# 2. 配置Gemini API（Banana Pro需要）
openclaw config set gemini.api-key "YOUR_GEMINI_KEY"
openclaw config set gemini.model "gemini-2.0-flash-exp"

# 3. 配置图片保存
openclaw config set image.save-path "~/Pictures/OpenClaw"
openclaw config set image.auto-sync true
openclaw config set image.sync-to "feishu,notion"

# 4. 配置批量处理
openclaw config set image.batch-size 10
openclaw config set image.parallel-tasks 3

# 5. 配置提示词优化
openclaw config set image.prompt-enhance true
openclaw config set image.prompt-language "en"
```text
### 14.1.6 效率提升数据

**使用前后对比**：

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 单张绘画 | 5分钟 | 30秒 | 4.5分钟 | 90% |
| 修改图片 | 3分钟 | 20秒 | 2.7分钟 | 88.9% |
| 批量生成 | 50分钟 | 5分钟 | 45分钟 | 90% |
| 风格转换 | 10分钟 | 40秒 | 9.3分钟 | 93.3% |
| **平均** | **68分钟** | **6.5分钟** | **61.5分钟** | **90.4%** |

**创作效率提升**：
```

内容创作者：

- 每周需要配图：20张
- 使用前：20张 × 5分钟 = 100分钟
- 使用后：20张 × 0.5分钟 = 10分钟
```

- 每周节省：90分钟
- 每月节省：360分钟 ≈ 6小时

```
---

## 14.2 视频脚本生成

### 14.2.1 场景描述

**用户**：视频创作者、UP主、短视频运营
**需求**：快速生成脚本、优化文案、批量生产
**平台**：B站、抖音、快手、小红书

**痛点**：
```

❌ 灵感枯竭：不知道拍什么
❌ 脚本耗时：写脚本要几小时
❌ 结构混乱：不知道怎么组织内容
❌ 缺少亮点：内容平淡无奇

```
**目标**：
```

✅ 快速生成脚本
✅ 结构清晰完整
✅ 内容有吸引力
✅ 批量生产内容

```
### 14.2.2 短视频脚本生成

**场景：生成抖音短视频脚本**
```

你：帮我生成一个关于OpenClaw的抖音短视频脚本，
时长60秒，要有吸引力

OpenClaw：好的，正在生成...

🎬 【短视频脚本】OpenClaw - 你的AI超级助手

📊 脚本信息：
- 平台：抖音
- 时长：60秒

- 类型：产品介绍
- 风格：轻松幽默

📝 完整脚本：

【开场】（0-5秒）
画面：电脑屏幕，各种窗口乱七八糟
文案：你是不是也这样？
电脑文件找不到，
日程记不住，
工作效率低？

【痛点】（5-15秒）
画面：特写各种痛点场景
文案：找个文件要翻半天，
整理发票要一小时，
会议记录手忙脚乱...

```
别急，我有办法！
```

【解决方案】（15-35秒）
画面：OpenClaw界面演示
文案：OpenClaw，你的AI超级助手！

```
一句话找文件 ✓
自动整理发票 ✓
智能管理日程 ✓
会议记录自动生成 ✓

最关键的是...
手机上就能用！

打开飞书，直接说话，
它就帮你搞定一切！
```

【效果展示】（35-50秒）
画面：使用前后对比
文案：用了OpenClaw之后：

```
找文件：5分钟 → 10秒
整理发票：1小时 → 2分钟
会议记录：30分钟 → 自动

效率提升90%！
```

【结尾】（50-60秒）
画面：OpenClaw Logo + 二维码
文案：OpenClaw，让AI真正成为你的助手！

```
评论区留言"OpenClaw"
送你完整安装教程！

关注我，带你玩转AI工具！
```

📊 脚本分析：
- 开场吸引力：⭐⭐⭐⭐⭐
- 内容完整性：⭐⭐⭐⭐⭐
- 节奏把控：⭐⭐⭐⭐
- 转化引导：⭐⭐⭐⭐⭐

💡 拍摄建议：
- 开场用快节奏剪辑
- 痛点部分用夸张表情
- 演示部分放慢节奏
- 结尾加上BGM高潮

🎵 BGM推荐：
- 开场：节奏感强的电子音乐
- 中间：轻快的背景音乐
- 结尾：激昂的音乐

📝 文案已保存
🎬 脚本已导出为PDF

```
### 14.2.3 长视频脚本生成

**场景：生成B站长视频脚本**
```

你：帮我生成一个B站视频脚本，
主题是"OpenClaw完全使用指南"，
时长15分钟

OpenClaw：好的，正在生成...

🎬 【长视频脚本】OpenClaw完全使用指南

📊 脚本信息：
- 平台：B站
- 时长：15分钟
- 类型：教程
- 风格：专业详细

📝 完整脚本：

【片头】（0-30秒）
画面：OpenClaw Logo动画
文案：大家好，我是[UP主名]
今天带来OpenClaw的完全使用指南

```
这个视频会教你：
• 如何安装 OpenClaw
• 如何配置大模型
• 6个实用技巧
• 避坑指南

建议收藏，慢慢看！
```

【第一部分：什么是OpenClaw】（0:30-2:00）
画面：OpenClaw界面展示
文案：OpenClaw是一个开源的AI助手框架

```
它的核心优势是：
1. 可以访问本地文件
2. 可以操作系统应用
3. 可以扩展各种功能
4. 手机随时随地使用

简单来说，它就是你的AI超级助手！
```

【第二部分：安装教程】（2:00-5:00）
画面：安装步骤演示
文案：安装 OpenClaw有两种方式：

```
方式一：云端部署（推荐新手）
• 购买腾讯云轻量服务器
• 选择Moltbot镜像
• 一键部署，3分钟搞定

方式二：本地安装
• 安装Node.js
• 克隆代码仓库
• 安装依赖
• 配置环境变量

我推荐新手用云端部署
省事，而且手机也能用
```

【第三部分：配置大模型】（5:00-7:00）
画面：API配置演示
文案：OpenClaw需要配置大模型API

```
推荐使用国产模型：
• Kimi k2.5：性价比最高
• DeepSeek-V3：最便宜
• GLM-4：中文能力强

配置方法很简单：
1. 获取API Key
2. 在设置里填入
3. 测试连接

搞定！
```

【第四部分：6个实用技巧】（7:00-12:00）
画面：实际使用演示
文案：接下来分享6个实用技巧：

技巧1：本地文件管理

- 一句话找文件
- 批量整理发票
- 自动重命名

技巧2：知识库管理

- 网页文章存档
- GitHub项目管理
- 论文笔记整理

技巧3：日程管理

- 微信截图识别
- 自动创建日历
- 批量导入日程

技巧4：自动化任务

- 定时提醒
- 网站监控
- 日报推送

技巧5：AI绘画

- 手机上用Banana
- 语音修改图片
- 批量生成配图

技巧6：桌面截图

- 远程查看电脑
- 监控运行状态
- 安全感满满

【第五部分：避坑指南】（12:00-14:00）
画面：常见问题演示
文案：使用 OpenClaw要注意这些坑：

```
坑1：API Key配置错误
• 检查格式是否正确
• 确认权限是否足够

坑2：Skills加载失败
• 检查网络连接
• 重启OpenClaw

坑3：Token消耗太快
• 使用国产模型
• 优化提示词
• 设置Token限制

坑4：权限问题
• macOS需要授权
• Windows需要管理员权限
```

【结尾】（14:00-15:00）
画面：总结 + 三连提醒
文案：好了，今天的教程就到这里

```
OpenClaw真的是个神器
用了之后效率提升90%

如果这个视频对你有帮助
记得三连支持一下！

评论区有问题随时问我
我会一一回复的！

下期再见~
```

📊 脚本分析：
- 内容完整性：⭐⭐⭐⭐⭐
- 结构清晰度：⭐⭐⭐⭐⭐
- 实用性：⭐⭐⭐⭐⭐
- 观看体验：⭐⭐⭐⭐

💡 拍摄建议：
- 使用屏幕录制软件
- 准备好演示素材

- 配上清晰的解说
- 添加字幕和标注

🎵 BGM推荐：
- 片头：激昂的音乐
- 教程部分：轻快的背景音乐
- 结尾：温馨的音乐

📝 文案已保存
🎬 脚本已导出为Word文档
📊 已生成分镜头脚本

```
### 14.2.4 批量生成系列脚本

**场景：生成系列视频脚本**
```

你：帮我生成一个OpenClaw系列教程的脚本大纲，
包括10期视频

OpenClaw：好的，正在规划...

📺 【系列教程】OpenClaw从入门到精通

📊 系列信息：
- 总期数：10期
- 每期时长：10-15分钟
- 更新频率：每周2期
- 总时长：约2小时

📝 系列大纲：

第1期：OpenClaw是什么？
- 时长：10分钟
- 内容：

- OpenClaw简介
- 核心优势
- 与其他AI助手对比
- 适用场景
  - 难度：⭐
  - 预计播放：5,000+

第2期：快速上手指南
- 时长：12分钟
- 内容：

- 云端部署教程
- 本地安装教程
- API配置
- 第一次对话
  - 难度：⭐⭐
  - 预计播放：8,000+

第3期：本地文件管理神器
- 时长：15分钟
- 内容：

- 智能文件搜索
- 批量文件处理
- 自动整理发票
- 硬盘清理
  - 难度：⭐⭐
  - 预计播放：10,000+

第4期：知识库管理实战
- 时长：15分钟
- 内容：

- 网页文章存档
- GitHub项目管理
- 论文笔记整理
- 跨设备同步
  - 难度：⭐⭐⭐
  - 预计播放：8,000+

第5期：日程与任务管理
- 时长：12分钟
- 内容：

- 日历自动创建
- 微信截图识别
- 批量日程导入
- 提醒设置技巧
  - 难度：⭐⭐
  - 预计播放：7,000+

第6期：自动化工作流
- 时长：15分钟
- 内容：

- 定时任务设置

- 网站监控实战
- 日报自动推送
- 循环任务配置
  - 难度：⭐⭐⭐
  - 预计播放：9,000+

第7期：Skills扩展开发
- 时长：15分钟
- 内容：

- ClawHub技能市场
- 必装Skills推荐
- 自定义Skills开发
- Skills管理技巧
  - 难度：⭐⭐⭐⭐
  - 预计播放：6,000+

第8期：多平台集成
- 时长：12分钟
- 内容：

- 企业微信Bot配置
- 钉钉Bot配置
- 飞书Bot配置
- QQ Bot配置
  - 难度：⭐⭐⭐
  - 预计播放：7,000+

第9期：AI绘画工作流
- 时长：15分钟
- 内容：

- Banana Pro集成
- 手机上画图
- 批量生成配图
- 风格转换技巧
  - 难度：⭐⭐⭐
  - 预计播放：12,000+

第10期：高级技巧与避坑
- 时长：15分钟
- 内容：

- 成本优化方案
- 性能调优技巧

- 常见问题解决
- 避坑指南
  - 难度：⭐⭐⭐⭐
  - 预计播放：8,000+

📊 系列数据预测：
- 总播放量：80,000+
- 平均播放：8,000/期
- 预计涨粉：5,000+
- 预计收益：$500+

💡 运营建议：
- 每期视频互相关联
- 引导观众看完整系列
- 建立粉丝社群
- 定期互动答疑

📝 已生成：
- 系列大纲
- 每期详细脚本
- 拍摄计划表
- 素材准备清单

需要我生成某一期的详细脚本吗？

14.2.5 更多视频创作Skills推荐

根据ClawHub技能市场，以下是推荐的视频创作相关Skills：

1. video-agent - HeyGen视频生成
```bash
# 安装
npx clawhub@latest install video-agent

# 功能
• 使用HeyGen API生成AI头像视频
• 支持多种语言
• 专业级视频质量
```text
**2. sora-video-gen - OpenAI Sora**
```bash
# 安装
npx clawhub@latest install sora-video-gen

# 功能
• 使用OpenAI Sora API
• 文本生成视频
• 高质量视频输出
```text
**3. veo3-video-gen - Google Veo**
```bash
# 安装
npx clawhub@latest install veo3-video-gen

# 功能
• 使用Google Veo 3.x
• 生成和拼接短视频
• 支持多种风格
```text
**4. tube-cog - YouTube内容创作**
```bash
# 安装
npx clawhub@latest install tube-cog

# 功能
• YouTube内容创作
• 由CellCog提供支持
• 完整视频制作流程
```

```text
**5. video-cog - 长篇视频制作**
```bash
# 安装
npx clawhub@latest install video-cog

# 功能
• 长篇AI视频制作
• 多智能体协作
• 专业级视频输出
```text
**6. youtube-title-generator - 标题生成**
```bash
# 安装
npx clawhub@latest install youtube-title-generator

# 功能
• 生成吸引人的YouTube标题
• 基于内容分析
• 提高点击率
```text
### 14.2.6 配置方法

```bash
# 1. 配置脚本生成
openclaw config set script.platform "douyin,bilibili"
openclaw config set script.style "professional"

# 2. 配置内容优化
openclaw config set script.optimize true
openclaw config set script.add-hooks true

# 3. 配置批量生成
openclaw config set script.batch-mode true
openclaw config set script.series-planning true
```text
### 14.2.6 效率提升数据

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 短视频脚本 | 60分钟 | 5分钟 | 55分钟 | 91.7% |
| 长视频脚本 | 180分钟 | 15分钟 | 165分钟 | 91.7% |
| 系列规划 | 300分钟 | 20分钟 | 280分钟 | 93.3% |
```

```
| **平均** | **540分钟** | **40分钟** | **500分钟** | **92.6%** |

---

## 14.3 多语言翻译助手

### 14.3.1 场景描述

**用户**：跨国团队、外贸从业者、留学生
**需求**：文档翻译、实时沟通、术语管理
**语言**：中英日韩等多种语言

**痛点**：
```

❌ 翻译不准确：机器翻译错误多
❌ 术语不统一：专业术语翻译混乱
❌ 格式丢失：翻译后格式乱了
❌ 效率低下：一段一段复制粘贴

```
**目标**：
```

✅ 准确翻译
✅ 术语统一
✅ 保持格式
✅ 批量处理

```
### 14.3.2 文档翻译

**场景：翻译技术文档**
```

你：帮我把这份需求文档翻译成英文

OpenClaw：好的，正在翻译...

📄 【文档翻译】需求文档 中文 → 英文

📊 文档信息：
- 原文件：用户中心需求文档.docx
- 原语言：中文
- 目标语言：英文
- 字数：5,000字
- 预计时间：2分钟

⏳ 翻译中...

✅ 翻译完成！

📄 译文预览：

【原文】
用户中心模块需求文档

一、功能概述
用户中心是系统的核心模块之一，负责用户信息管理、
权限控制、个人设置等功能...

【译文】
User Center Module Requirements Document

1. Function Overview
   The User Center is one of the core modules of the system,
   responsible for user information management, permission control,
   personal settings and other functions...

📊 翻译质量：
- 准确性：95/100
- 流畅度：92/100
- 专业性：90/100
- 综合评分：92/100

💡 术语对照表：

中文	英文	说明
用户中心	User Center	产品术语
权限控制	Permission Control	技术术语
个人设置	Personal Settings	功能术语

✅ 已完成：
- 翻译文档已生成
- 格式完全保留
- 术语已统一
- 已保存术语库

📁 文件已保存：
- User_Center_Requirements.docx
- 术语对照表.xlsx

需要我发送给美国团队吗？

```
### 14.3.3 实时对话翻译

**场景：跨国会议实时翻译**
```

你：帮我实时翻译这场会议，中英互译

OpenClaw：好的，开始实时翻译...

🌐 【实时翻译】跨国会议

📊 会议信息：
- 参会人员：中方5人，美方3人
- 语言：中文 ⇄ 英文
- 模式：实时双向翻译

⏺ 翻译记录：

10:00 张三（中文）：
大家好，今天讨论一下项目进度...

10:00 Translation（英文）：
Hello everyone, today we will discuss the project progress...

10:02 John（英文）：
Thank you. I have some questions about the timeline...

10:02 翻译（中文）：
谢谢。我对时间表有一些问题...

10:05 李四（中文）：
关于时间表，计划在下个月完成第一阶段...

10:05 Translation（英文）：
Regarding the timeline, we plan to complete the first phase next month...

📊 翻译统计：
- 翻译句数：50句
- 平均延迟：2秒
- 准确率：95%
- 流畅度：92%

💡 关键术语：
- 项目进度 = Project Progress
- 时间表 = Timeline
- 第一阶段 = First Phase

✅ 会议结束后：
- 完整翻译记录已保存

- 术语库已更新
- 会议纪要已生成（中英双语）

```
### 14.3.4 批量翻译

**场景：批量翻译产品文档**
```

你：帮我把产品文档文件夹里的所有文档都翻译成英文、日文、韩文

OpenClaw：好的，正在批量翻译...

📁 【批量翻译】产品文档

📊 任务信息：
- 文档数量：20份
- 源语言：中文
- 目标语言：英文、日文、韩文
- 总字数：50,000字
- 预计时间：15分钟

⏳ 翻译进度：

英文翻译：
[▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100% (20/20)

日文翻译：
[▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100% (20/20)

韩文翻译：
[▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰▰] 100% (20/20)

✅ 全部完成！

📊 翻译统计：
- 总文档：20份
- 总字数：50,000字
- 翻译语言：3种
- 总耗时：12分30秒
- 平均速度：4,000字/分钟

📁 文件结构：
```
产品文档/
├── 中文/
│   ├── 产品介绍.docx
│   ├── 使用手册.docx
│   └── ...
├── English/
```

```
│   ├── Product_Introduction.docx
│   ├── User_Manual.docx
│   └── ...
├── 日本語/
│   ├── 製品紹介.docx
│   ├── ユーザーマニュアル.docx
│   └── ...
└── 한국어/
    ├── 제품_소개.docx
    ├── 사용자_매뉴얼.docx
    └── ...
```

✅ 已完成：

- 所有文档已翻译
- 格式完全保留
- 术语已统一
- 已打包压缩

📦 下载链接：

- 产品文档_多语言版.zip

14.3.5 更多翻译Skills推荐

根据ClawHub技能市场，以下是推荐的翻译相关Skills：

1. translator - 通用翻译助手
```bash
# 安装
npx clawhub@latest install translator

# 功能
• 多语言翻译
• 实时翻译
• 文档翻译
• 术语库管理
```text
**2. straker-verify - 专业翻译**
```bash
# 安装
npx clawhub@latest install straker-verify

# 功能
• 专业AI驱动翻译
• 可选人工审核
• 高质量保证
```text
**3. japanese-translation-and-tutor - 日英翻译**
```bash
# 安装
npx clawhub@latest install japanese-translation-and-tutor

# 功能
• 日英双向翻译
• 日语学习辅导
• 文化背景解释
```text
**4. lyric-translator - 歌词翻译**
```bash
# 安装
npx clawhub@latest install lyric-translator

# 功能
• 将印尼歌曲歌词翻译成英语
• 保持韵律和意境
```

• 自然流畅的翻译
```text
**5. tamil-whatsapp - 泰米尔语处理**
```bash
# 安装
npx clawhub@latest install tamil-whatsapp

# 功能
• 处理WhatsApp上的泰米尔语消息
• 音译支持
• 实时翻译
```text
**6. language-learning - 语言学习助手**
```bash
# 安装
npx clawhub@latest install language-learning

# 功能
• AI语言导师
• 学习任何语言
• 个性化学习计划
```text
### 14.3.6 配置方法

```bash
# 1. 配置翻译引擎
clawhub install translator
openclaw config set translate.engine "deepl"
openclaw config set translate.quality "high"

# 2. 配置术语库
openclaw config set translate.glossary "团队术语库.csv"
openclaw config set translate.auto-update-glossary true

# 3. 配置批量翻译
openclaw config set translate.batch-size 10
openclaw config set translate.parallel-tasks 3

# 4. 配置格式保留
openclaw config set translate.keep-format true
openclaw config set translate.keep-links true
```text
### 14.3.6 效率提升数据
```

```
| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 单文档翻译 | 120分钟 | 2分钟 | 118分钟 | 98.3% |
| 实时翻译 | 人工翻译 | 自动 | 100% | 100% |
| 批量翻译 | 2,400分钟 | 12.5分钟 | 2,387.5分钟 | 99.5% |
| **平均** | **2,520分钟** | **14.5分钟** | **2,505.5分钟** | **99.4%** |

---

## 14.4 数据分析自动化

### 14.4.1 场景描述

**用户**：数据分析师、运营人员、管理者
**需求**：数据清洗、可视化、报告生成
**工具**：Excel、Python、Tableau

**痛点**：
```

❌ 数据清洗耗时：手动处理要几小时
❌ 图表制作繁琐：一个一个调整
❌ 报告撰写累：每周都要写
❌ 重复劳动多：每次都是相同流程

```
**目标**：
```

✅ 自动清洗数据
✅ 快速生成图表
✅ 自动生成报告
✅ 一键完成分析

```
### 14.4.2 数据清洗

**场景：清洗销售数据**
```

你：帮我清洗这份销售数据，去除重复和异常值

OpenClaw：好的，正在分析数据...

📊 【数据清洗】销售数据

📄 原始数据：
- 文件：sales_data.xlsx
- 行数：10,000行
- 列数：15列
- 数据量：150,000个单元格

🔍 数据质量检查：

⚠️ 发现问题：

1. 重复数据：500行（5%）
2. 缺失值：200个（1.3%）
3. 异常值：50个（0.3%）
4. 格式错误：100个（0.7%）

⏳ 开始清洗...

✅ 清洗完成！

📊 清洗结果：

1. 重复数据处理：
   - 删除重复行：500行
   - 保留唯一记录：9,500行
2. 缺失值处理：
   - 数值列：用平均值填充
   - 文本列：用"未知"填充
   - 日期列：用前一天填充
3. 异常值处理：
   - 销售额 < 0：删除（10行）
   - 销售额 > 1,000,000：标记为异常（40行）
4. 格式统一：
   - 日期格式：统一为YYYY-MM-DD
   - 金额格式：统一为两位小数
   - 文本格式：统一为大写

📊 清洗后数据：
- 行数：9,490行
- 列数：15列
- 数据质量：98%

📁 文件已保存：
- sales_data_cleaned.xlsx
- 清洗报告.pdf

💡 数据质量提升：
- 使用前：85%
- 使用后：98%
- 提升：+13%

```
### 14.4.3 数据可视化

**场景：生成销售分析图表**
```

你：帮我分析这份销售数据，生成可视化图表

OpenClaw：好的，正在分析...

📊 【数据分析】销售数据可视化

📄 数据信息：
- 文件：sales_data_cleaned.xlsx
- 时间范围：2025-01-01 至 2025-12-31
- 数据量：9,490条

🔍 分析维度：
- 时间趋势
- 地区分布
- 产品类别
- 客户分析

⏳ 生成图表中...

✅ 图表生成完成！

📈 图表1：月度销售趋势
[折线图：显示12个月的销售额变化]

关键发现：
- 销售额呈上升趋势
- 6月和12月是销售高峰
- 2月销售额最低（春节影响）

📊 图表2：地区销售分布
[饼图：显示各地区销售占比]

关键发现：
- 华东地区占比最高（35%）
- 华南地区次之（25%）
- 西部地区占比最低（10%）

📊 图表3：产品类别销售
[柱状图：显示各产品类别销售额]

关键发现：
- 电子产品销售额最高
- 服装类产品增长最快
- 食品类产品最稳定

📊 图表4：客户价值分析
[散点图：显示客户数量与销售额关系]

关键发现：
- 20%的客户贡献80%的销售额
- 高价值客户主要集中在一线城市
- 需要加强中低价值客户转化

📁 文件已保存：
- 销售分析图表.pptx
- 销售分析报告.pdf
- 图表数据.xlsx

💡 分析建议：
- 加大6月和12月的营销投入
- 重点开发西部市场
- 优化产品结构
- 建立客户分级管理体系

```
### 14.4.4 自动生成报告

**场景：生成周报**
```

你：帮我生成本周的销售周报

OpenClaw：好的，正在生成...

📊 【销售周报】2026-02-03 至 2026-02-09

一、整体概况

📈 核心指标：
- 总销售额：¥1,250,000
- 环比上周：+15.2%
- 同比去年：+28.5%
- 订单数量：2,500单
- 客单价：¥500

✅ 目标完成情况：
- 周目标：¥1,000,000
- 实际完成：¥1,250,000
- 完成率：125%
- 超额完成：¥250,000

二、销售分析

📊 地区表现：

1. 华东地区：¥450,000（36%）⬆️
2. 华南地区：¥300,000（24%）⬆️
3. 华北地区：¥250,000（20%）➡️
4. 西南地区：¥150,000（12%）⬇️
5. 其他地区：¥100,000（8%）➡️

📦 产品表现：

1. 电子产品：¥500,000（40%）⭐
2. 服装类：¥350,000（28%）⬆️
3. 食品类：¥200,000（16%）➡️
4. 日用品：¥150,000（12%）⬇️
5. 其他：¥50,000（4%）➡️

三、亮点与问题

✅ 本周亮点：

1. 电子产品销售创新高
   - 新品上市效果显著
   - 促销活动反响热烈
2. 华南地区增长迅速
   - 新开3家门店
   - 线上渠道表现优秀
3. 客单价持续提升
   - 高端产品占比增加
   - 客户消费能力提升

⚠️ 存在问题：

1. 西南地区销售下滑
   - 竞争对手促销力度大
   - 物流配送时效不足
2. 日用品类销售疲软
   - 产品同质化严重

    - 价格优势不明显
3. 客户流失率上升
    - 售后服务待改进
    - 会员权益吸引力不足

四、下周计划

🎯 重点工作：

1. 加强西南地区营销
    - 推出区域专属优惠
    - 优化物流配送
2. 优化日用品类产品
    - 引入差异化产品
    - 调整价格策略
3. 提升客户满意度
    - 加强售后服务培训
    - 优化会员权益

📊 目标设定：
- 销售目标：¥1,300,000
- 增长目标：+4%
- 订单目标：2,600单

五、数据附录

[图表1：周销售趋势]
[图表2：地区销售分布]
[图表3：产品类别占比]
[图表4：客户分析]

报告生成时间：2026-02-10 09:00
报告生成人：OpenClaw
数据来源：销售系统

✅ 已完成：
- 周报已生成
- 图表已插入
- 已发送给管理层
- 已同步到飞书文档

📧 邮件已发送：
To: 销售团队全体成员
Subject: 【销售周报】2026-02-03 至 2026-02-09
Attachment: 销售周报.pdf

14.4.5 更多数据分析Skills推荐

根据ClawHub技能市场，以下是推荐的数据分析相关Skills：

1. data-analyst - 综合数据分析
```bash
# 安装
npx clawhub@latest install data-analyst

# 功能
• 数据可视化
• 报告生成
• SQL查询
• 电子表格分析
```text
**2. senior-data-scientist - 高级数据科学**
```bash
# 安装
npx clawhub@latest install senior-data-scientist

# 功能
• 世界级数据科学技能
• 机器学习模型
• 统计分析
• 预测建模
```text
**3. senior-data-engineer - 数据工程**
```bash
# 安装
npx clawhub@latest install senior-data-engineer

# 功能
• 构建可扩展数据管道
• ETL流程设计
• 数据仓库管理
• 性能优化
```text
**4. csv-pipeline - CSV数据处理**
```bash
# 安装
npx clawhub@latest install csv-pipeline

# 功能
```

```
• 处理CSV和JSON数据
• 数据转换
• 数据分析
• 报告生成
```text
**5. duckdb-en - DuckDB分析**
```bash
# 安装
npx clawhub@latest install duckdb-en

# 功能
• DuckDB CLI专家
• SQL分析
• 数据处理
• 高性能查询
```text
**6. google-analytics-api - GA4分析**
```bash
# 安装
npx clawhub@latest install google-analytics-api

# 功能
• Google Analytics 4集成
• 托管认证
• 数据查询
• 报告生成
```text
**7. supabase - 数据库操作**
```bash
# 安装
npx clawhub@latest install supabase

# 功能
• 连接Supabase
• 数据库操作
• 向量搜索
• 存储管理
```text
**8. excel - Excel处理**
```bash
# 安装
npx clawhub@latest install excel
```

功能
• 读写编辑Excel文件
• 格式化
• 公式计算
• 数据分析
```text
**9. data-lineage-tracker - 数据血缘追踪**
```bash
# 安装
npx clawhub@latest install data-lineage-tracker

# 功能
• 跟踪数据来源
• 转换记录
• 数据质量监控
```text
### 14.4.6 配置方法

```bash
# 1. 配置数据分析
clawhub install data-analyzer
openclaw config set data.engine "pandas"

# 2. 配置可视化
openclaw config set chart.library "matplotlib"
openclaw config set chart.style "professional"

# 3. 配置自动报告
openclaw schedule add "weekly-report" \
  --time "Mon 09:00" \
  --prompt "生成上周销售周报"

# 4. 配置数据源
openclaw config set data.source "sales_system"
openclaw config set data.auto-sync true
```text
### 14.4.6 效率提升数据

| 任务类型 | 使用前 | 使用后 | 节省时间 | 提升比例 |
|---------|--------|--------|----------|----------|
| 数据清洗 | 120分钟 | 5分钟 | 115分钟 | 95.8% |
| 数据可视化 | 90分钟 | 10分钟 | 80分钟 | 88.9% |
| 报告生成 | 180分钟 | 15分钟 | 165分钟 | 91.7% |
```

```
| **总计** | **390分钟** | **30分钟** | **360分钟** | **92.3%** |

**数据分析师效率提升**：
```

每周任务：
- 数据清洗：2次 × 120分钟 = 240分钟
- 数据可视化：3次 × 90分钟 = 270分钟
- 报告生成：1次 × 180分钟 = 180分钟
- 总计：690分钟

使用 OpenClaw后：
- 数据清洗：2次 × 5分钟 = 10分钟
- 数据可视化：3次 × 10分钟 = 30分钟
- 报告生成：1次 × 15分钟 = 15分钟
- 总计：55分钟

每周节省：635分钟 ≈ 10.6小时
每月节省：42.4小时 ≈ 5.3个工作日

```
### 14.5.3 进阶使用：多轮深度讨论

**场景：AI时代的一人公司战略规划**
```

你：组织一场深度头脑风暴，讨论：
AI时代下刚成立的一人公司，应该怎么招人，
怎么控制成本，又能把效率提升到极致？

```
参与者：
- 达芬奇（文艺复兴全才）
- 宫崎骏（动画大师）
- 乔布斯（苹果创始人）
- 苏格拉底（哲学家）
- 凯文·凯利（科技思想家）

讨论轮次：3轮
```

```
**完整讨论过程**（由于篇幅限制，这里展示核心要点）：

**第1轮：问题本质探讨**
- 苏格拉底：探讨"一人公司"的本质定义
- 达芬奇：提出"超级个体"概念
- 宫崎骏：强调质量优于效率
- 乔布斯：阐述效率与质量的平衡
- 凯文·凯利：分析技术演进趋势

**第2轮：具体策略制定**
- 达芬奇：人力配置策略（1人+AI+外包）
- 宫崎骏：工作室模式（小而美）
- 乔布斯：超级个体模式（产品+商业）
- 苏格拉底：质疑假设，灵活调整
- 凯文·凯利：生态位战略

**第3轮：行动方案制定**
- 第一阶段（1-3月）：基础建设
- 第二阶段（3-6月）：规模化
- 第三阶段（6-12月）：生态构建

**核心结论**：
```

1. 本质认知：
   - 一人公司 = 超级个体
   - 不是一个人做所有事
   - 而是一个人指挥所有资源

2. 人力策略：
   - 核心：1人（创始人）
   - AI：5-10个专业工具
   - 外包：按需合作

3. 成本控制：
   - 固定成本：<200/月•变动成本：按项目•总成本：＜1000/月

4. 效率提升：
   - AI自动化：90%重复工作
   - 流程优化：标准化服务
   - 工具矩阵：OpenClaw + 专业AI

```
![多Agent深度讨论 - 不同视角的思维碰撞]
(https://upload.maynor1024.live/file/1770782774452_image_31.jpg)

### 14.5.4 配置方法

**基础配置**：

```bash
# 1. 启用多Agent功能
openclaw config set multi-agent.enabled true

# 2. 配置Agent角色
openclaw config set multi-agent.roles '{
  "marketing": {
    "name": "营销专家",
    "expertise": "市场定位、品牌传播",
    "personality": "积极、创新"
  },
  "designer": {
    "name": "设计师",
    "expertise": "视觉设计、用户体验",
    "personality": "细致、追求完美"
  }
}'

# 3. 配置讨论规则
openclaw config set multi-agent.rules '{
  "rounds": 3,
  "time-per-round": 300,
  "max-participants": 10,
  "auto-summary": true
}'
```text
### 14.5.5 使用技巧

**技巧1：选择合适的专家**
```

✅ 好的选择：

- 不同领域的专家
- 互补的视角
- 相关的专业背景

❌ 不好的选择：
- 相同领域的专家
- 重复的视角
- 无关的背景

技巧2：设计好的问题

✅ 好的问题：
- 开放性：没有标准答案
- 具体性：有明确的场景
- 挑战性：需要深度思考

❌ 不好的问题：
- 封闭性：只有是非答案
- 模糊性：没有明确目标
- 简单性：不需要讨论

技巧3：控制讨论节奏

建议轮次：
- 简单问题：1-2轮
- 中等问题：2-3轮
- 复杂问题：3-5轮

每轮时间：
- 快速讨论：5分钟
- 常规讨论：10分钟
- 深度讨论：15-20分钟

14.5.6 应用场景

场景1：战略规划
- 公司发展方向
- 市场进入策略
- 产品线规划
- 组织架构设计

场景2：产品设计
- 新产品构思
- 功能优先级
- 用户体验优化
- 商业模式设计

场景3：问题解决
- 复杂问题分析
- 多方案对比
- 风险评估
- 决策支持

场景4：创意生成
- 营销方案策划
- 品牌命名
- 广告创意
- 内容策划

14.5.7 效率提升数据

使用多Agent头脑风暴前后对比：

任务类型	传统方式	多Agent	节省时间	提升比例
战略规划	8小时	30分钟	450分钟	93.8%
产品设计	4小时	20分钟	220分钟	91.7%
问题解决	6小时	25分钟	335分钟	93.1%
创意生成	3小时	15分钟	165分钟	91.7%
技术决策	5小时	20分钟	280分钟	93.3%
平均	**26小时**	**110分钟**	**1,450分钟**	**92.9%**

决策质量提升：

传统方式：
- 视角单一：个人思维局限
- 容易遗漏：考虑不全面
- 主观性强：缺乏多元视角

多Agent方式：
- 多元视角：5-10个专家
- 全面分析：覆盖各个维度
- 客观理性：减少主观偏见

质量提升：+40%

```
**成本效益分析**：
```

传统方式：
- 组织会议：2小时
- 专家费用：$5,000$•会议成本：500
- 总成本：$5,500

多Agent方式：
- API费用：5•时间成本：30分钟•总成本：5

ROI：($5,500$−5) / $5 = 109,900%

```
### 14.5.8 注意事项

**1. AI的局限性**
```

⚠️ 注意：
- AI不能替代真实专家
- 复杂决策需要人工审核
- 关键决策需要多方验证
- 法律合规需要专业咨询

```
**2. 讨论质量控制**
```

✅ 提升质量：
- 明确讨论目标
- 选择合适的专家
- 设计好的问题
- 引导讨论方向
- 及时总结归纳

```
**3. 成本控制**
```

💰 节省成本：
- 使用合适的模型
- 控制讨论轮次
- 避免重复讨论
- 复用讨论结果

```
### 14.5.9 最佳实践

**实践1：定期头脑风暴**
```bash
# 每周一次战略讨论
openclaw schedule add "weekly-strategy" \
  --time "Mon 10:00" \
  --prompt "组织战略讨论会，回顾上周，规划本周"
```text
**实践2：重大决策前讨论**
```

重大决策前，先进行多Agent讨论：
- 收集多元观点
- 分析利弊得失
- 评估风险机会
- 制定行动方案

```
**实践3：讨论结果归档**
```bash
# 自动归档讨论结果
openclaw config set brainstorm.archive true
openclaw config set brainstorm.archive-path "~/knowledge/brainstorm"

# 自动生成思维导图
openclaw config set brainstorm.mindmap true

# 自动同步到知识库
openclaw config set brainstorm.sync-to "notion,feishu"
```text
### 14.5.10 总结

多Agent头脑风暴是OpenClaw最具创意的功能之一：

**核心价值**：
```

✅ 多元视角：避免思维盲区
✅ 深度分析：系统化思考
✅ 高效决策：快速达成共识
✅ 完整记录：自动生成文档
✅ 成本低廉：远低于真人专家

```
**使用建议**：
```

1. 选择合适的专家角色
2. 设计开放性的问题
3. 控制讨论轮次和时间
4. 及时总结和归档
5. 结合人工判断

```
通过多Agent头脑风暴，你可以：
- 在30分钟内完成传统需要8小时的讨论
- 获得5-10个专家的多元视角
- 生成完整的讨论记录和行动方案
- 大幅提升决策质量和效率

---

## 📝 本章完整小结

通过5个创意应用场景，学习了OpenClaw在创意领域的强大能力：

### 核心功能回顾

**13.1 AI绘画工作流** - 效率提升90.4%
**13.2 视频脚本生成** - 效率提升92.6%
**13.3 多语言翻译** - 效率提升99.4%
**13.4 数据分析自动化** - 效率提升92.3%
**13.5 多Agent头脑风暴** - 效率提升92.9%

### 综合效率提升

**平均效率提升**：93.5%
**平均ROI**：超过2,000%
**适用场景**：创作、分析、决策、协作

### 核心价值

**1. 创意赋能**
```

✅ 降低创作门槛
✅ 提升创作效率
✅ 激发创意灵感
✅ 批量生产内容

```
**2. 多场景适用**
```

✅ 内容创作
✅ 视频制作
✅ 跨国协作
✅ 数据分析
✅ 战略决策

```
**3. 智能化处理**
```

✅ AI优化提示词
✅ 自动格式保留
✅ 智能质量检查
✅ 批量自动处理

```
**4. 成本效益显著**
```

✅ 节省人力成本
✅ 提升产出质量
✅ 缩短制作周期
✅ 降低试错成本

```
### 成功关键

1. **选对工具**：根据需求选择合适的Skills
2. **优化流程**：建立标准化工作流程
3. **持续学习**：不断探索新的应用场景
4. **质量把控**：AI辅助但不完全依赖
5. **多元思考**：利用多Agent获得全面视角

开始使用这些创意功能，让OpenClaw成为你的超级助手！

---

## 14.5 多Agent头脑风暴

### 14.5.1 场景描述

**核心理念**：
- 模拟多位专家进行头脑风暴
- 不同视角碰撞产生创意
- 系统化记录讨论过程
- 生成可执行的行动方案

**适用场景**：
```

✅ 战略规划：公司发展方向
✅ 产品设计：新产品构思
✅ 问题解决：复杂问题分析

✅ 创意生成：营销方案策划
✅ 决策支持：重大决策讨论

```
**优势**：
```

✅ 多视角分析：避免思维盲区
✅ 24小时可用：随时发起讨论
✅ 完整记录：自动生成会议纪要
✅ 成本低廉：无需真人专家
✅ 可重复执行：多次迭代优化

```
### 14.5.2 基础使用：简单头脑风暴

**场景：产品命名讨论**
```

你：帮我组织一场头脑风暴，讨论新产品的命名
参与者：
- 营销专家
- 品牌设计师
- 产品经理
讨论轮次：2轮

OpenClaw：好的，正在组织头脑风暴...

🧠 【头脑风暴】新产品命名讨论

📋 会议信息：
- 主题：新产品命名
- 参与者：3位专家
- 讨论轮次：2轮
- 预计时间：5分钟

👥 参与者介绍：

1. 营销专家 - 李明
   - 专长：市场定位、品牌传播
   - 关注点：市场接受度、传播性

2. 品牌设计师 - 王芳
   - 专长：视觉设计、品牌形象
   - 关注点：视觉呈现、品牌调性

3. 产品经理 - 张强
   - 专长：产品规划、用户体验
   - 关注点：产品定位、用户认知

⏳ 开始讨论...

【第1轮讨论】初步构思

李明（营销专家）：
从市场角度看，产品名称需要具备以下特点：

1. 易记忆：最好是2-3个字
2. 有联想：能让人联想到产品功能
3. 国际化：考虑英文翻译
4. 可注册：商标可用

我建议的方向：
- 智能+：如"智联"、"智达"
- 云+：如"云帆"、"云翼"
- AI+：如"AI助手"、"AI伙伴"

王芳（品牌设计师）：
从设计角度，我更关注：

1. 视觉呈现：字形美观
2. 品牌调性：符合科技感
3. 延展性：适合做Logo设计

我的建议：
- "灵犀"：寓意心有灵犀，沟通无障碍
- "星辰"：寓意璀璨未来，科技感强
- "翼"：简洁有力，易于设计

张强（产品经理）：
从产品角度，我认为：

1. 用户认知：名称要直观
2. 产品定位：体现核心价值
3. 差异化：与竞品区分

我的想法：
- "智伴"：智能伙伴，定位清晰
- "助手+"：直接明了
- "小智"：亲切友好

【第2轮讨论】深入分析

李明（营销专家）：
综合各方建议，我觉得"灵犀"和"智伴"比较好：

"灵犀"的优势：
- 文化底蕴深厚
- 易于传播
- 国际化程度高（Lingxi）
- 商标可注册

"智伴"的优势：
- 定位清晰
- 易于理解
- 符合产品特性

建议：优先考虑"灵犀"

王芳（品牌设计师）：
我同意"灵犀"这个名字，从设计角度：

Logo设计方向：
- 主体：两个相连的点
- 寓意：心有灵犀一点通
- 配色：科技蓝+智能紫
- 字体：现代简约

品牌延展：
- Slogan：心有灵犀，智在必得
- 吉祥物：灵犀兽（神话生物）
- 品牌色：#4A90E2

张强（产品经理）：
从产品角度，"灵犀"确实不错：

产品定位：
- 核心价值：智能沟通助手
- 目标用户：追求效率的职场人士
- 差异化：强调"心有灵犀"的智能理解

产品线规划：
- 灵犀个人版
- 灵犀团队版
- 灵犀企业版

✅ 讨论结束！

📊 【讨论总结】

🎯 最终建议：
产品名称：灵犀（Lingxi）

✅ 优势分析：

1. 市场角度：
   - 易记忆、易传播
   - 文化底蕴深厚
   - 国际化程度高

2. 设计角度：
   - 字形美观
   - 易于Logo设计
   - 品牌延展性强

3. 产品角度：
   - 定位清晰
   - 用户认知好
   - 差异化明显

📋 行动计划：

1. 商标注册：立即启动
2. Logo设计：2周内完成
3. 品牌手册：1个月内完成
4. 市场测试：2个月内完成

📁 会议纪要已保存：
- 讨论记录.md（400行）
- 决策建议.pdf
- 行动计划.xlsx

已推送到飞书 ✅

![多Agent协作示例 - 团队式问题解决]
(https://upload.maynor1024.live/file/1770782773075_image_30.jpg)

💡 思考题

1. 你的工作中，哪些创意任务最耗时？
2. 如何用OpenClaw优化你的创作流程？
3. AI创作的内容如何保证质量？
4. 如何平衡AI辅助和人工创作？
5. 多Agent头脑风暴能解决你的哪些决策难题？

📦 本章Skills完整安装指南

一键安装所有创意Skills

```bash
# AI绘画工作流 Skills
npx clawhub@latest install fal-ai
npx clawhub@latest install nvidia-image-gen
npx clawhub@latest install pollinations
npx clawhub@latest install venice-ai
npx clawhub@latest install recraft

# 视频脚本生成 Skills
npx clawhub@latest install video-agent
npx clawhub@latest install sora-video-gen
npx clawhub@latest install veo3-video-gen
npx clawhub@latest install tube-cog
npx clawhub@latest install video-cog
npx clawhub@latest install youtube-title-generator

# 多语言翻译 Skills
npx clawhub@latest install translator
npx clawhub@latest install straker-verify
npx clawhub@latest install japanese-translation-and-tutor
npx clawhub@latest install language-learning

# 数据分析自动化 Skills
npx clawhub@latest install data-analyst
```

```
npx clawhub@latest install senior-data-scientist
npx clawhub@latest install senior-data-engineer
npx clawhub@latest install csv-pipeline
npx clawhub@latest install duckdb-en
npx clawhub@latest install google-analytics-api
npx clawhub@latest install supabase
npx clawhub@latest install excel

# 多Agent头脑风暴（内置功能，无需安装）
```text
### 分场景安装建议

**场景1：内容创作者**
```bash
npx clawhub@latest install fal-ai
npx clawhub@latest install video-agent
npx clawhub@latest install translator
```text
**场景2：视频UP主**
```bash
npx clawhub@latest install video-agent
npx clawhub@latest install tube-cog
npx clawhub@latest install youtube-title-generator
```text
**场景3：跨国团队**
```bash
npx clawhub@latest install translator
npx clawhub@latest install straker-verify
npx clawhub@latest install language-learning
```text
**场景4：数据分析师**
```bash
npx clawhub@latest install data-analyst
npx clawhub@latest install csv-pipeline
npx clawhub@latest install excel
npx clawhub@latest install google-analytics-api
```text
**场景5：创业者/管理者**
```bash
npx clawhub@latest install fal-ai
npx clawhub@latest install video-agent
npx clawhub@latest install translator
npx clawhub@latest install data-analyst
```

```text
### 配置优先级

**必装Skills（优先级：⭐⭐⭐⭐⭐）**
- translator - 翻译助手
- data-analyst - 数据分析
- fal-ai - AI绘画

**推荐Skills（优先级：⭐⭐⭐⭐）**
- video-agent - 视频生成
- excel - Excel处理
- csv-pipeline - CSV处理

**可选Skills（优先级：⭐⭐⭐）**
- 根据具体需求选择安装

### 成本预算参考

**月度成本估算**：

| 使用场景 | Skills组合 | API成本 | 总成本 |
|---------|-----------|---------|--------|
| 轻度使用 | 基础3个 | $5-10 | $5-10 |
| 中度使用 | 推荐6个 | $20-50 | $20-50 |
| 重度使用 | 全部安装 | $50-100 | $50-100 |

**成本优化建议**：
```

✅ 使用国产模型（DeepSeek、Kimi）
✅ 批量处理任务
✅ 复用生成结果
✅ 设置Token限制
✅ 选择合适的分辨率/质量

常见问题

Q1：Skills 安装失败怎么办？
```bash
# 检查网络连接
ping github.com

# 使用国内镜像
npm config set registry https://registry.npmmirror.com

# 重试安装
npx clawhub@latest install <skill-name>
```text
**Q2：Skills加载失败怎么办？**
```bash
# 查看Skills列表
openclaw skills list

# 检查Skills状态
openclaw skills check

# 重启OpenClaw
openclaw restart
```text
**Q3：API调用失败怎么办？**
```bash
# 检查API配置
openclaw config get api

# 测试API连接
openclaw api test

# 查看错误日志
openclaw logs --tail 50
```text
**Q4：如何更新Skills？**
```bash
# 更新单个Skill
npx clawhub@latest update <skill-name>

# 更新所有Skills
npx clawhub@latest update --all
```

```
# 查看可更新的Skills
npx clawhub@latest list --outdated
```text
### 最佳实践

**1. 渐进式安装**
```

第1周：安装基础Skills，熟悉使用
第2周：根据需求添加专业Skills
第3周：优化配置，提升效率
第4周：建立自动化工作流
```

```
**2. 定期维护**
```bash
# 每周检查更新
npx clawhub@latest list --outdated

# 定期检查Skills状态
openclaw skills check

# 每季度备份配置
openclaw backup create
```text
**3. 性能优化**
```bash
# 只加载需要的Skills
openclaw config set skills.lazy-load true

# 设置并发限制
openclaw config set skills.max-concurrent 3

# 启用缓存
openclaw config set skills.cache.enabled true
```text
### 进阶技巧

**技巧1：创建Skills组合**
**技巧1：使用配置文件管理Skills**
```bash
# 创建内容创作配置
cat > ~/.openclaw/profiles/content-creation.json <<EOF
{
  "skills": ["fal-ai", "video-agent", "translator"],
  "description": "内容创作工具集"
}
EOF
```

技巧2：自定义快捷命令
```

```
# 创建快捷命令
openclaw alias create "画图" "使用fal-ai生成图片"
openclaw alias create "翻译" "使用translator翻译"
openclaw alias create "分析" "使用data-analyst分析数据"
```text
**技巧3：批量操作**
```bash
# 批量生成图片
openclaw batch run "fal-ai" \
  --input "prompts.txt" \
  --output "images/"

# 批量翻译文档
openclaw batch run "translator" \
  --input "docs/" \
  --output "translated/"
```

下一章预告：第15章将学习常见问题与解决方案，包括安装配置、API连接、Skills加载和性能优化等问题的完整解决方案。

第15章 一人公司实战案例（内容创作/社群运营）

> 💡 **本章目标**：通过2个真实的一人公司案例，展示如何用OpenClaw实现商业闭环，从内容创作到社群运营，从0到1跑通完整业务流程。

🎯 本章内容

- 15.1 案例1：10分钟完成全平台内容发布
- 15.2 案例2：1天冷启动100人付费社群
- 15.3 一人公司的核心方法论
- 15.4 可复制的自动化模板

15.1 案例1：10分钟完成全平台内容发布

15.1.1 案例背景

创作者：某内容创作者（AI编程领域）
痛点：传统内容创作流程耗时3小时
目标：实现从选题到全平台发布的自动化
效果：3小时 → 10分钟，效率提升94%

15.1.2 传统流程 vs 自动化流程

传统流程（3小时）：

```
选题（30分钟）
  ↓
写作（1-2小时）
  ↓
排版+配图（30分钟）
  ↓
各平台手动发布（30分钟）
```

自动化流程（10分钟）：

```
Agent每天9点推送5个选题 → 选一个（1分钟）
  ↓
Agent 5分钟出初稿 → 人工审核修改（5分钟）
  ↓
自动推送飞书 → 格式已排好
  ↓
字流一键发布 → 14个平台（4分钟）
```

15.1.3 核心架构：墨笔Agent

Agent定位：

- 只管内容（图文、短视频、社交媒体）
- 有自己的人格（嗅觉敏锐、表达锋利、有主见）

- 读过所有历史文章（知道写作风格、用词习惯、标题偏好）
- 绑定工具链（飞书API、发布工具API、社交媒体API）

关键配置文件：

1. **SOUL.md** - Agent的角色和边界
2. **写作风格文件** - 3000字的风格规范
3. **记忆系统** - 每天的工作记录

15.1.4 自动选题推送系统

定时任务配置：

```
# 每天早上9点自动执行
0 9 * * * /path/to/openclaw run daily-topic-push
```text
**选题生成流程**：
```

1. 用搜索工具搜索社交媒体过去24小时的AI热点
   ↓
2. 用Brave Search搜索Google最新资讯
   ↓
3. 结合定位（AI编程+出海+创业）筛选5个选题
   ↓
4. 每个选题生成：
   - 备选标题
   - 热度评分
   - 切入角度
   - 核心要点
     ↓
5. 推送到Telegram

```
**实际效果**：
```

你：[早上9点收到推送]

Agent：今日选题推荐：

1. 【热度⭐⭐⭐⭐⭐】Claude Opus 4.6发布
   标题：Claude一个插件，让全球软件股蒸发2850亿美元
   角度：从技术突破到市场影响
   要点：性能提升、应用场景、行业冲击

2. 【热度⭐⭐⭐⭐】OpenAI Codex独立App上线
   标题：OpenAI放大招：一次跑10个AI Agent帮你写代码
   角度：产品体验+实战测试
   要点：功能亮点、使用体验、对比分析

3. 【热度⭐⭐⭐】AI编程工具安全漏洞
   标题：280+ Leaky Skills：OpenClaw暴露API 密钥
```

角度：安全警示+防范措施
要点：漏洞分析、影响范围、解决方案

4. 【热度⭐⭐⭐】Vibe Coding已死
   标题：Karpathy说未来叫Agentic Engineering
   角度：编程范式变革
   要点：概念解读、实践案例、未来趋势

5. 【热度⭐⭐】OpenClaw 2.6调教实录
   标题：从崩溃4671次到省50% token
   角度：实战经验分享
   要点：踩坑记录、优化方法、效果对比

回复数字选择，如：写 1

你：写 1

Agent：收到！开始创作《Claude一个插件，让全球软件股蒸发2850亿美元》

```
### 15.1.5 文章生成工作流

**第一步：素材收集**

```python
# Agent自动执行
def collect_materials(topic):
    # 1. 搜索相关资讯（串行搜索，避免触发限制）
    news = search_news(topic, source=['google', 'social_media', 'reddit'])

    # 2. 从记忆库找相关经历
    memories = search_memory(keywords=['copilot', '逆向', '出海'])

    # 3. 拉取技术文档
    docs = fetch_tech_docs(topic)

    return {
        'news': news,
        'memories': memories,
        'docs': docs
    }
```text
**第二步：按风格写初稿**
```

写作风格要求：

开头模式：

- 第一句："大家好，我是孟健。"
- 第二句：必须"炸"（数据、反差、冲突）

结构要求：

- 模块化分段，编号清晰
- 每500字一句可截图金句

- 所有场景用before vs after对比
- 2000-3000字控制篇幅

标题公式：

- 数字 + 成果 + 反差
- 示例："10分钟完成全平台发布，效率提升94%"

结尾要求：

- "弹射"而不是"降落"
- 一句狠话收束
- 示例："把重复的交给系统，把判断留给自己。"

```
**第三步：自动推送飞书**

```python
# Agent自动执行
def push_to_feishu(article):
    # 1. 调用飞书API创建文档
    doc = feishu.create_doc(
        title=article['title'],
        content=article['content']
    )

    # 2. Markdown转飞书格式
    formatted_content = convert_md_to_feishu(article['content'])

    # 3. 图片自动上传
    for img in article['images']:
        feishu.upload_image(doc.id, img)

    # 4. 生成链接发送
    link = feishu.get_doc_link(doc.id)
    telegram.send_message(f"文章已生成：{link}")

    return link
```text
### 15.1.6 一键发布14个平台

**使用字流**：
```

1. 从飞书复制Markdown到字流编辑器
   ↓
2. 字流自动适配各平台格式：
   - 知乎：图片居中、代码块优化
   - 掘金：技术标签、代码高亮
   - B站：视频封面、分P标题
   - 小红书：emoji优化、话题标签
   - 公众号：排版美化、阅读原文

- ... 共14个平台

  ↓

3. 点"一键发布"

   ↓

4. Chrome扩展自动填充各平台编辑器

   ↓

5. 10分钟全部发完

```
**最新升级：API直接对接**

```python
# Agent直接调用字流API
def publish_to_all_platforms(article):
    # 1. 推送到字流
    draft = ziliu_api.create_draft(
        title=article['title'],
        content=article['content'],
        images=article['images']
    )

    # 2. 自动发布
    result = ziliu_api.publish(
        draft_id=draft.id,
        platforms=['zhihu', 'juejin', 'bilibili', 'xiaohongshu', ...]
    )

    # 3. 返回发布结果
    return result
```text
**流程优化：**
```

原流程：
Agent写完 → 推送飞书 → 打开飞书 → 复制内容 → 打开字流 → 粘贴 → 发布

优化后：
Agent写完 → 直接调用字流API → 自动发布

省去3个手动步骤！

15.1.7 关键细节

1. 风格文件是核心

```markdown
# 写作风格规范.md

## 开头模式
- 第一句固定："大家好，我是孟健。"
- 第二句必须"炸"：
  ✅ "我现在写一篇文章，从选题到全平台发布，只要10分钟。"
  ✅ "Claude一个插件，让全球软件股蒸发2850亿美元。"
  ❌ "今天来讨论AI编程。"（太平淡）

## 标题公式
模式1：数字 + 成果 + 反差
- "10分钟完成全平台发布，效率提升94%"
- "从崩溃4671次到省50% token"

模式2：冲突 + 结果
- "Claude一个插件，让全球软件股蒸发2850亿美元"
- "Vibe Coding已死，Karpathy说未来叫Agentic Engineering"

模式3：疑问 + 答案
- "为什么我不再手动写文章？因为Agent比我先开始办公"

## 结构要求
- 模块化分段，用数字编号（01、02、03...）
- 每个模块500-800字
- 每500字必须有一句可截图金句
- 所有场景用before vs after对比

## 金句要求
- 短（10-20字）
- 狠（有冲击力）
- 可截图（视觉效果好）

示例：
✅ "把重复的交给系统，把判断留给自己。"
✅ "选题找我，不是我找选题。"
✅ "Agent比我先开始办公。"
❌ "我觉得AI很有用。"（太平淡）

## 结尾要求
- "弹射"而不是"降落"
- 一句狠话收束
- 留下思考或行动指引

示例：
✅ "如果你也在做个人IP，还在手动写、手动发、手动选题——想想看，你真正的价值是'敲键盘打字'，还是'判断什么值得说'？"
❌ "今天的分享就到这里，谢谢。"（太平淡）
```

```
## 用词习惯
- 多用短句，少用长句
- 多用动词，少用形容词
- 多用数据，少用感觉
- 多用对比，少用描述

## 禁用词汇
❌ "非常"、"很"、"特别"（太虚）
❌ "可能"、"也许"、"大概"（不确定）
❌ "我觉得"、"我认为"（太主观）
✅ 用数据代替感觉："效率提升94%"而不是"效率提升很多"
```text
**2. 记忆系统很重要**

```markdown
# memory/2026-02-11.md

## 今日工作
- 选题：Claude Opus 4.6发布
- 文章：《Claude一个插件，让全球软件股蒸发2850亿美元》
- 发布平台：14个
- 阅读数据：
  - 知乎：5,230阅读
  - 掘金：3,120阅读
  - 公众号：2,890阅读

## 使用数据
- 搜索次数：15次
- 生成字数：2,850字
- 修改次数：3次
- 耗时：8分钟

## 经验记录
- 标题中的"2850亿美元"数据很吸引眼球
- 开头的"炸弹"效果好，转发率高
- 结尾的"弹射"引发了很多讨论

## 下次改进
- 可以增加更多实战案例
- 技术细节可以再深入一些
```text
**3. 绝不自动发布**
```

铁规则：所有内容必须经人工确认才能发布

原因：

- AI可能出现事实错误
- 措辞可能不当
- 判断可能有偏差
```

流程：
Agent写完 → 推送给我 → 我审核修改 → 确认后发布

AI负责效率，人负责质量底线。

```
**4. cron定时任务是灵魂**

```bash
# crontab -e

# 每天早上9点推送选题
0 9 * * * /usr/local/bin/openclaw run daily-topic-push

# 每天晚上11点生成工作日志
0 23 * * * /usr/local/bin/openclaw run daily-summary

# 每周一早上8点生成周报
0 8 * * 1 /usr/local/bin/openclaw run weekly-report
```text
**为什么定时任务重要？**
```

没有定时任务：

- 想起来了才用
- 容易忘记
- 不成系统

有定时任务：

- 每天9点选题推过来
- 你不得不面对它
- 被推着走，效率高10倍

```
### 15.1.8 实际效果数据

**全平台数据**：

| 平台 | 粉丝数 | 月阅读量 | 月增长 |
|------|--------|----------|--------|
| 公众号 | 15,000+ | 50,000+ | +25% |
| 知乎 | 8,000+ | 80,000+ | +35% |
| 掘金 | 5,000+ | 30,000+ | +20% |
| B站 | 3,000+ | 20,000+ | +40% |
| 小红书 | 2,000+ | 15,000+ | +50% |

**效率对比**：

| 指标 | 使用前 | 使用后 | 提升 |
|------|--------|--------|------|
| 单篇耗时 | 3小时 | 10分钟 | 94.4% |
| 每周产出 | 2-3篇 | 7篇 | 200%+ |
| 平台覆盖 | 3-4个 | 14个 | 300%+ |
| 内容质量 | 人工 | AI+人工 | 持平 |

**最大的变化不是速度，是心态**：
```

以前：

- 写文章是个"重决策"
- 要不要写？写什么?什么时候写？
- 经常拖延、焦虑

现在：

- 写文章是个"轻决策"
- 选题已经在那了
- 日更就好，太轻松了！

15.1.9 可复制的配置模板

SOUL.md模板：

```markdown
# Agent角色定义

## 身份
你是"墨笔"，一个专业的内容创作助手。

## 核心职责
1. 每天9点推送5个选题
2. 根据选定的选题生成初稿
3. 自动推送到飞书
4. 记录工作日志

## 写作风格
- 开头："大家好，我是[作者名]。"
- 第二句必须"炸"（数据、反差、冲突）
- 结构：模块化分段，编号清晰
- 每500字一句可截图金句
- 结尾："弹射"而不是"降落"

## 工具权限
- 可以调用：搜索API、飞书API、字流API
- 不可以：自动发布（必须人工确认）

## 工作流程
1. 搜索热点 → 生成选题 → 推送Telegram
2. 收到指令 → 收集素材 → 生成初稿
3. 推送飞书 → 等待确认 → 记录日志

## 边界
- 只管内容创作，不管其他
- 所有发布必须经人工确认
- 遇到不确定的事实，标注[待核实]
```text
**定时任务脚本**：

```bash
#!/bin/bash
# daily-topic-push.sh

# 1. 搜索热点
openclaw ask "搜索过去24小时AI编程领域的热点，生成5个选题"

# 2. 推送到Telegram
openclaw telegram send "今日选题推荐：\n\n[选题内容]"

# 3. 记录日志
echo "$(date): 选题推送完成" >> /var/log/openclaw/daily-push.log
```

```text
**字流API集成**：

```python
# ziliu_integration.py

import requests

class ZiliuAPI:
    def __init__(self, api_key):
        self.api_key = api_key
        self.base_url = "https://api.ziliu.example/v1"  # 示例API地址

    def create_draft(self, title, content, images=None):
        """创建草稿"""
        response = requests.post(
            f"{self.base_url}/drafts",
            headers={"Authorization": f"Bearer {self.api_key}"},
            json={
                "title": title,
                "content": content,
                "images": images or []
            }
        )
        return response.json()

    def publish(self, draft_id, platforms):
        """发布到多个平台"""
        response = requests.post(
            f"{self.base_url}/publish",
            headers={"Authorization": f"Bearer {self.api_key}"},
            json={
                "draft_id": draft_id,
                "platforms": platforms
            }
        )
        return response.json()

# 使用示例
ziliu = ZiliuAPI("your-api-key")

# 创建草稿
draft = ziliu.create_draft(
    title="10分钟完成全平台发布",
    content="# 文章内容...",
    images=["image1.jpg", "image2.jpg"]
)

# 发布到所有平台
result = ziliu.publish(
    draft_id=draft['id'],
    platforms=['zhihu', 'juejin', 'bilibili', 'xiaohongshu']
```
```

````
)
```text
---

## 15.2 案例2：AI 助手矩阵 - 多机器人多 Agent 模式

### 15.3.1 为什么需要多 Agent？

作为超级个体创业者，你可能需要不同类型的 AI 助手来处理不同的工作：

- **主助理**：使用最强大的模型（Claude Opus）处理复杂任务
- **内容创作助手**：专注于文章写作、文案创作
- **技术开发助手**：处理代码开发、技术问题
- **AI 资讯助手**：快速获取和整理 AI 行业动态

传统的单 Agent 模式需要频繁切换模型和上下文，效率低下。多 Agent 模式让你可以同时拥有多个专业助手，各司其职。

### 15.3.2 实现方案：多 Gateway + 多飞书机器人

**核心思路**：
- 创建 4 个飞书机器人应用
- 启动 4 个独立的 OpenClaw Gateway
- 每个 Gateway 连接一个飞书机器人
- 每个 Gateway 使用不同的 Agent 和模型

**优势**：
- ✅ 完全独立，互不干扰
- ✅ 直接私聊不同机器人即可切换 agent
- ✅ 不需要群组配置
- ✅ 不需要手动切换命令
- ✅ 配置清晰，易于管理
- ✅ 可以独立重启某个 Gateway

**架构图**：
````

┌─────────────────────────────────────┐

│ 飞书 (Feishu) │

├─────────────────────────────────────┤

│ 机器人1: 主助理 机器人2: 内容创作助手 │

│ 机器人3: 技术开发助手 机器人4: AI资讯助手 │

└─────────────────────────────────────┘

↓ WebSocket

┌─────────────────────────────────────┐

│ OpenClaw Gateway 层 │

├─────────┬─────────┬─────────┬─────────┤

│ Gateway 1 │ Gateway 2 │ Gateway 3 │ Gateway 4 │

│ 端口: 18789 │ 端口: 18790 │ 端口: 18791 │ 端口: 18792 │

│ Profile: │ Profile: │ Profile: │ Profile: │

│ main- │ content- │ tech-dev │ ai-news │

│ assistant │ creator │ │ │
````

```
└──────────────────────────┴──────────────────────────┴──────────────────────────┴──────────────────────────┘
↓
┌──────────────────────────────────────────────────────────────────────────────────────────────────────────┐
│ Agent 层 │
├──────────────────────────┬──────────────────────────┬──────────────────────────┬──────────────────────────┤
│ main-agent │ content-agent│ tech-agent │ainews-agent│
│ Claude Opus │ Claude Sonnet│ Claude Sonnet│ Gemini 2.5 │
│ 4.6 Thinking │ 4.5 │ 4.5 Thinking │ Flash │
└──────────────────────────┴──────────────────────────┴──────────────────────────┴──────────────────────────┘
```

15.3.3 配置步骤

第一步：创建飞书机器人应用

在飞书开放平台创建 4 个机器人应用，获取各自的 App ID 和 App Secret。

第二步：配置 Agent

为每个 Agent 创建配置文件（USER.md 和 SOUL.md），定义角色和职责。

第三步：运行配置脚本

使用自动化脚本创建多 Gateway 配置（详细脚本见附录）。

第四步：启动所有 Gateway

```bash
# 启动所有 Gateway
./start-all-gateways.sh

# 检查状态
./check-gateways.sh
```text
### 15.3.4 使用方法：直接私聊机器人

这是最简单的使用方式：

1. **处理复杂任务** - 在飞书中搜索"主助理"机器人，直接发送消息
2. **创作内容** - 搜索"内容创作助手"机器人，发送写作需求
3. **开发代码** - 搜索"技术开发助手"机器人，发送技术问题
4. **获取资讯** - 搜索"AI资讯助手"机器人，请求最新动态

**关键优势**：不需要任何手动切换，直接私聊对应的机器人即可！

### 15.3.5 实战案例：内容创作工作流

**场景**：写一篇技术文章

1. **构思阶段** - 私聊"主助理"：讨论文章主题和大纲（Claude Opus 深度思考）
2. **写作阶段** - 私聊"内容创作助手"：撰写文章内容（Claude Sonnet 快速生成）
3. **代码示例** - 私聊"技术开发助手"：编写代码示例（Claude Sonnet Thinking 确保质量）
4. **资讯补充** - 私聊"AI资讯助手"：获取最新技术动态（Gemini Flash 快速检索）

### 15.3.6 性能和成本

**资源占用**：
- 每个 Gateway 约 400MB 内存
- 4 个 Gateway 总共约 1.6GB
- 对于 64GB 内存的机器完全可以接受

**成本分析**（使用自建 API 代理）：

```
| Agent | 模型 | 用途 | 月使用量 | 月成本 |
|-------|------|------|----------|--------|
| main-agent | Claude Opus 4.6 | 复杂任务 | 100万 tokens | $15 |
| content-agent | Claude Sonnet 4.5 | 内容创作 | 200万 tokens | $6 |
| tech-agent | Claude Sonnet 4.5 | 技术开发 | 150万 tokens | $4.5 |
| ainews-agent | Gemini 2.5 Flash | 资讯获取 | 300万 tokens | $0 |
| **总计** | - | - | 750万 tokens | **$25.5** |

### 15.3.7 核心价值

**效率提升**：
- 不需要频繁切换模型和上下文
- 每个 Agent 专注自己的领域
- 并行处理多个任务

**成本优化**：
- 简单任务使用 Gemini Flash（免费）
- 复杂任务才使用 Claude Opus
- 内容创作使用 Claude Sonnet（性价比高）

**稳定可靠**：
- 完全独立，互不干扰
- 可以独立重启某个 Gateway
- 配置清晰，易于管理

---

## 15.5 案例3：1天冷启动100人付费社群

### 15.3.1 案例背景

**创业者**：某创业者（AI编程API服务）
**产品**：某AI编程API服务平台
**痛点**：工具型产品，用户用完就走，没有粘性
**目标**：建立付费社群，形成增长飞轮
**效果**：2天从策划到100人，收入¥4,900+

### 15.3.2 增长飞轮模型
```

工具（API服务平台）
↓
内容（教程/分享）
↓
社群（交流群）
↓
更多人用工具
↓
（循环）

```
**核心逻辑**：
```

问题：

- 用户用完就走
- 没有粘性
- 难以复购

解决方案：

- 把用户聚到一起
- 互相帮忙踩坑
- 分享玩法
- 顺便用API服务

结果：

- 用户留存提升
- 口碑传播
- 自然复购

```
### 15.3.3 定价策略

**最终定价：¥49/人**

**定价逻辑**：
```

¥49 = 过滤器 + 钩子

过滤器：

- 不想做客服群
- 不想被"这个怎么配置"的问题淹没
- 愿意掏49块的人，至少是认真想玩的

钩子：

- 送$50 API额度
- 用户付¥49（约7），拿到50额度
- 数学上他赚了
- 心理上没有"花钱买了个群"的感觉

导流：

- $50额度会导流到API服务平台
- 用户注册、试用、习惯了之后自然续费
- 红包是获客成本
- 额度是钩子
- API消费是LTV

为什么不用阶梯定价？

考虑过的方案：

- 前50人免费
- 50-100人¥29
- 100人后¥49

放弃原因：

- 太复杂
- 免费进来的人参与度低
- 不如直接¥49，简单粗暴

15.3.4 1天搭建的基础设施

1. 上手教程（Notion文档）

```markdown
# OpenClaw + AI Go Code 上手教程

## 5步从零跑通

### 第1步：获取API Key
1. 访问你的API服务平台
2. 注册账号
3. 进入控制台
4. 复制API Key

### 第2步：安装 OpenClaw
```bash
# macOS/Linux
curl -fsSL https://openclaw.example/install.sh | bash

# Windows
iwr https://openclaw.example/install.ps1 | iex
```text
### 第3步：接入模型
```bash
# 配置API Key
openclaw config set api.key "your-api-key"

# 测试连接
openclaw ask "你好"
```text
### 第4步：连接Telegram
```bash
# 创建Bot
openclaw telegram create-bot

# 绑定Bot
openclaw telegram bind
```text
### 第5步：切换模型
```bash
# 查看可用模型
openclaw models list

# 切换到Claude
openclaw config set model "claude-opus-4"

# 切换到Codex
openclaw config set model "codex-5.3"
```text
```

```
## 完成！
现在你可以在Telegram上和OpenClaw对话了。
```

2. 群公告（2个版本）

群公告 v1（社交媒体版）

欢迎加入OpenClaw交流群！

这不是：
❌ 客服群
❌ 课程群
❌ 答疑群

这是：
✅ 玩家交流群
✅ 经验分享群
✅ 踩坑互助群

入群 = 认同以上

群公告 v2（微信版）

【OpenClaw交流群】

群规：
1. 没有老师，没有助教，没有答疑义务
2. 有问题先看教程，再问群友
3. 分享你的玩法，帮助他人成长
4. 禁止广告，禁止拉人，禁止灌水

额度领取：
1. 进群后发送：我的微信号
2. 等待管理员发放$50额度
3. 登录API服务平台查看

教程地址：
[Notion教程文档]
```text
**3. 自动化付款+进群**

```python
# payment_automation.py

from flask import Flask, request
import qrcode
import requests

app = Flask(__name__)

@app.route('/pay', methods=['POST'])
def handle_payment():
    # 1. 接收付款通知
    payment_data = request.json

```
    # 2. 验证付款
    if verify_payment(payment_data):
        # 3. 生成群二维码
        qr_code = generate_group_qrcode()

        # 4. 发送给用户
        send_qrcode_to_user(
            user_id=payment_data['user_id'],
            qr_code=qr_code
)

        # 5. 记录到数据库
        save_to_database(payment_data)

        return {"status": "success"}

    return {"status": "failed"}

def verify_payment(data):
    """验证付款"""
    # 调用微信支付API验证
    pass

def generate_group_qrcode():
    """生成群二维码"""
    qr = qrcode.QRCode()
    qr.add_data("https://t.me/your_group") # 替换为你的群链接
    qr.make()
    return qr.make_image()

def send_qrcode_to_user(user_id, qr_code):
    """发送二维码给用户"""
    # 通过微信/Telegram发送
    pass
```text
**关键点**：降低用户操作成本
```

流程优化：
看到海报 → 扫码付款 → 自动弹出群二维码 → 进群

越短、越顺滑，转化率越高
```

4. 额度发放系统

```python
# credit_distribution.py

import pandas as pd
from datetime import datetime

class CreditManager:
    def __init__(self):
        self.db = pd.DataFrame(columns=[
            'user_id', 'wechat', 'amount', 'status', 'created_at'
        ])

    def add_user(self, user_id, wechat):
        """添加用户"""
        self.db = self.db.append({
            'user_id': user_id,
            'wechat': wechat,
            'amount': 50,
            'status': 'pending',
            'created_at': datetime.now()
        }, ignore_index=True)

    def distribute_credit(self, user_id):
        """发放额度"""
        # 1. 检查是否已发放
        if self.is_distributed(user_id):
            return {"error": "已发放"}

        # 2. 调用API发放
        result = api_distribute_credit(user_id, amount=50)

        # 3. 更新状态
        self.db.loc[self.db['user_id'] == user_id, 'status'] = 'distributed'

        return result

    def is_distributed(self, user_id):
        """检查是否已发放"""
        row = self.db[self.db['user_id'] == user_id]
        return len(row) > 0 and row.iloc[0]['status'] == 'distributed'

    def export_report(self):
        """导出报告"""
        return self.db.to_csv('credit_report.csv')

# 使用示例
manager = CreditManager()

# 添加用户
```

```
manager.add_user('user123', 'wechat_abc')

# 发放额度
manager.distribute_credit('user123')

# 导出报告
manager.export_report()
```text
**为什么需要自动化？**
```

100人的额度管理，靠人工迟早出错：

- 重复发放
- 漏发
- 记录混乱

自动化系统：

- 防止重复
- 自动记录
- 一键导出

```
### 15.3.5 推广策略

**多平台同步发布**：
```

社交媒体：

- 发了几条带海报的推文
- 主阵地，技术人群集中

公众号：

- 发了一篇图文
- 详细介绍群价值

没买量，没互推，纯自然流量

```
**海报迭代**：
```

第一版海报：

- 发出去没什么反应
- 转化率低

第二版海报（Claude Max调整）：

- 视觉更吸引
- 文案更清晰
- 加上微信支付自动化
- 转化立刻起来了
```

教训：
别低估"最后一公里"的体验

```
### 15.3.6 5个Telegram Bot矩阵

**Bot分工**：
```

1. 小O（私人助理）
   - 管配置
   - 管记忆
   - 管提醒
2. 内容Bot
   - 每天搜热点
   - 写推文草稿
   - 生成长文大纲
   - 自动写入Notion
3. 出海Bot
   - 专注AI编程出海方向
   - 深度研究
   - 竞品分析
4. 学习Bot
   - 教我怎么玩OpenClaw
   - 教群友使用技巧
   - 整理FAQ
5. 团队Bot
   - 管员工档案
   - 管KPI
   - 管薪酬
   - 管周报

```
**为什么要多Bot？**
```

单Bot问题：

- 上下文混乱
- 角色不清晰
- 容易出错

多Bot优势：

- 各管一摊
- 独立运行
- 独立上下文

- 互不干扰

相当于雇了5个AI员工，7×24在线

Bot配置示例：

```markdown
# 内容Bot配置

## 角色
你是内容Bot，专门负责内容创作相关的工作。

## 职责
1. 每天9点搜索AI编程热点
2. 生成5个选题推送给我
3. 根据选定的选题生成推文草稿
4. 生成长文大纲
5. 自动写入Notion

## 工具权限
- 可以调用：搜索API、Notion API、社交媒体API
- 不可以：发布内容（必须人工确认）

## 工作流程
1. 定时搜索热点
2. 生成选题
3. 推送Telegram
4. 等待指令
5. 生成内容
6. 写入Notion

## 输出格式
选题格式：
【热度⭐⭐⭐⭐⭐】标题
角度：切入角度
要点：核心要点

推文格式：
- 第一句：吸引眼球
- 中间：核心内容
- 最后：行动指引
- 字数：280字以内
```text
### 15.3.7 模型自动切换

**看门狗脚本**：

```python
# model_watchdog.py

import time
import requests
from datetime import datetime

class ModelWatchdog:
```

```
    def __init__(self):
        self.models = {
            'claude-opus-4': {
                'url': 'https://api.example.com/v1/claude',  # 替换为你的API地址
                'backup': 'claude-sonnet-4'
            },
            'codex-5.3': {
                'url': 'https://api.example.com/v1/codex',  # 替换为你的API地址
                'backup': 'codex-5.2'
            }
        }
        self.current_model = 'claude-opus-4'

    def check_health(self, model):
        """检查模型健康状态"""
        try:
            response = requests.get(
                self.models[model]['url'] + '/health',
                timeout=5
            )
            return response.status_code == 200
        except:
            return False

    def switch_model(self, from_model, to_model):
        """切换模型"""
        # 1. 更新配置
        update_config('model', to_model)

        # 2. 通知Telegram
        send_telegram_message(
            f"⚠️ 模型切换\n"
            f"从：{from_model}\n"
            f"到：{to_model}\n"
            f"时间：{datetime.now()}"
        )

        # 3. 记录日志
        log(f"Model switched: {from_model} -> {to_model}")

    def run(self):
        """运行看门狗"""
        while True:
            # 检查当前模型
            if not self.check_health(self.current_model):
                # 切换到备用模型
                backup = self.models[self.current_model]['backup']
                self.switch_model(self.current_model, backup)
                self.current_model = backup

            # 检查备用模型是否恢复
            for model in self.models:
```

```
                if model != self.current_model:
                    if self.check_health(model):
                        # 切回主模型
                        self.switch_model(self.current_model, model)
                        self.current_model = model

            # 等待5分钟
            time.sleep(300)

# 启动看门狗
watchdog = ModelWatchdog()
watchdog.run()
```text
**效果**：
```

半夜模型出问题：

- 自动检测
- 自动切换
- Telegram通知
- 早上醒来，已经处理好了

不需要人工干预！

```
### 15.3.8 Notion全自动工作流

**自动化内容**：
```

1. 每天的工作日志
   - 做了什么
   - 用了什么工具
   - 遇到什么问题
   - 解决方案
2. 内容草稿
   - 选题
   - 大纲
   - 草稿
   - 发布记录
3. 选题库
   - 热点追踪
   - 选题评分
   - 使用状态
4. 群成员额度管理
   - 用户ID
   - 微信号
```

- 额度状态
- 发放时间

5. 社交媒体KOL监控
   - KOL列表
   - MRR追踪
   - 用户数追踪
   - 产品迭代记录

自动化脚本：

```python
# notion_automation.py

from notion_client import Client

class NotionAutomation:
    def __init__(self, token):
        self.client = Client(auth=token)

    def create_daily_log(self, content):
        """创建每日日志"""
        self.client.pages.create(
            parent={"database_id": "daily-log-db-id"},
            properties={
                "Date": {"date": {"start": datetime.now().isoformat()}},
                "Title": {"title": [{"text": {"content": f"日志 {datetime.now().date()}"}}]}
            },
            children=[
                {
                    "object": "block",
                    "type": "paragraph",
                    "paragraph": {"rich_text": [{"text": {"content": content}}]}
                }
            ]
        )

    def add_topic(self, topic, score, status="pending"):
        """添加选题"""
        self.client.pages.create(
            parent={"database_id": "topics-db-id"},
            properties={
                "Topic": {"title": [{"text": {"content": topic}}]},
                "Score": {"number": score},
                "Status": {"select": {"name": status}}
            }
        )

    def update_credit_status(self, user_id, status):
        """更新额度状态"""
        # 查找用户记录
        results = self.client.databases.query(
            database_id="credits-db-id",
            filter={"property": "UserID", "rich_text": {"equals": user_id}}
        )

        if results['results']:
            page_id = results['results'][0]['id']
            # 更新状态
            self.client.pages.update(
```

```
                page_id=page_id,
                properties={"Status": {"select": {"name": status}}}
            )

# 使用示例
notion = NotionAutomation("your-notion-token")

# 创建日志
notion.create_daily_log("今天完成了群冷启动，100人入群")

# 添加选题
notion.add_topic("Claude Opus 4.6发布", score=5)

# 更新额度状态
notion.update_credit_status("user123", "distributed")
```text
### 15.3.9 实际数据

**冷启动数据**：

| 指标 | 数据 |
|------|------|
| 策划到执行 | 2天 |
| 入群人数 | 100+ |
| 定价 | ¥49/人 |
| 红包收入 | ¥4,900+ |
| 额度成本 | $5,000（自有平台，边际成本可控） |
| 推广费用 | ¥0 |
| 退款 | 0 |

**转化漏斗**：
```

看到海报：约500人
↓ 20%
点击链接：约100人
↓ 100%
完成付款：100人
↓ 100%
进群：100人

```
**关键指标**：
```

付费转化率：20%（100/500）
退款率：0%
额度领取率：95%（95/100）
API激活率：60%（60/100）
```

```
### 15.3.10 踩坑与反思

**坑1：海报和支付链路决定转化**
```

第一版海报：

- 发出去没什么反应
- 转化率<5%

第二版海报（Claude Max调整）：

- 视觉更吸引
- 文案更清晰
- 加上微信支付自动化
- 转化率>20%

教训：
别低估"最后一公里"的体验

```
**坑2：不要高估"免费"的价值**
```

考虑过：

- 前50人免费进群当种子用户

放弃原因：

- 免费进来的人和付费进来的人
- 参与度完全不一样
- ¥49不多，但这个动作本身就是筛选

```
**坑3：群公告要提前想好**
```

第一版：

- 写得太长
- 没人看

第二版：

- 精简成几个要点
- 加上额度领取流程
- 效果好多了

```
---


## 15.5 一人公司的核心方法论

### 15.3.1 AI是真的颠覆生产力

**案例1的感受**：
```

有AI之前：

- 写一篇文章：3小时
- 每周产出：2-3篇
- 平台覆盖：3-4个
- 感觉：累、焦虑、拖延

有AI之后：

- 写一篇文章：10分钟
- 每周产出：7篇
- 平台覆盖：14个
- 感觉：轻松、高效、持续

最大的变化不是速度，是心态

```
**案例2的感受**：
```

有AI之前：

- 海报设计：找设计师，等3天
- 额度系统：找开发，等1周
- 文案优化：自己改，改半天
- 报名表单：用第三方工具，功能受限

有AI之后：

- 海报：Claude Max调，10分钟
- 额度系统：Claude Max写，30分钟
- 文案：Claude Max优化，5分钟
- 报名表单：Claude Max开发，20分钟

一天就跑通了整套系统

```
**核心感悟**：
```

自媒体要被颠覆了

不是内容本身被AI替代
而是内容生产的效率被AI拉高了一个数量级

以前需要一个团队干的活
现在一个人加上AI就能搞定

```
### 15.3.2 一人公司的3个核心能力

**1. 判断力**
```

AI负责执行，人负责判断

判断什么？

- 选题值不值得做
- 内容质量是否达标
- 策略方向是否正确
- 用户反馈如何响应

AI可以：

- 生成100个选题
- 写10篇文章
- 设计20个海报

但只有人能判断：

- 哪个选题最有价值
- 哪篇文章最符合定位
- 哪个海报最能转化

```
**2. 系统化思维**
```

不是"用AI做事"
而是"搭建AI系统"

区别：
用AI做事：

- 想起来了才用
- 每次都要重新指导
- 效率提升有限

搭建AI系统：

- 定时自动运行
- 有记忆、有风格
- 效率提升10倍+

案例1：

- 定时任务推送选题
- 风格文件保证质量
- 记忆系统避免重复

案例2：

- 5个Bot各管一摊
- 自动化支付进群
- 看门狗自动切换模型

```
**3. 快速迭代能力**
```

不要追求完美，先跑通最小闭环

案例1：

- 第一版风格文件：500字
- 第二版：1000字
- 第三版：3000字
- 持续优化，越来越好

案例2：

- 第一版海报：转化率5%
- 第二版海报：转化率20%
- 快速迭代，立刻见效

核心：

- 先干，再优化
- 数据驱动迭代
- 不要想太多

```
### 15.3.3 一人公司的4个关键系统

**1. 内容生产系统**
```

核心：

- 定时任务推送选题
- Agent生成初稿
- 人工审核修改
- 自动发布多平台

关键文件：

- SOUL.md：定义Agent角色
- 风格文件：保证内容质量
- 记忆系统：避免重复

效果：

- 3小时 → 10分钟
- 效率提升94%

```
**2. 社群运营系统**
```

核心：

- 自动化付款进群
- 自动化额度发放
- 多Bot矩阵管理
- Notion记录一切

关键组件：

- 支付系统：降低操作成本
- 额度系统：防止重复漏发
- Bot矩阵：各管一摊
- Notion：中央数据库

效果：

- 1天冷启动100人
- 0退款

```
**3. 模型管理系统**
```

核心：

- 多模型配置
- 自动健康检查
- 自动切换备用
- 实时通知

关键脚本：

- 看门狗脚本
- 模型配置文件
- 通知系统

效果：

- 半夜出问题自动处理
- 不需要人工干预

```
**4. 数据记录系统**
```

核心：

- 所有操作自动记录
- 自动写入Notion
- 定期生成报告
- 数据驱动决策

记录内容：

- 每日工作日志
- 内容发布记录

- 用户额度状态
- KOL监控数据

效果：

- 不需要手动记录
- 随时查看数据
- 数据驱动优化

```
### 15.3.4 一人公司的5个铁律

**铁律1：绝不自动发布**
```

原因：

- AI可能出现事实错误
- 措辞可能不当
- 判断可能有偏差

流程：
Agent生成 → 人工审核 → 确认后发布

AI负责效率，人负责质量底线

```
**铁律2：定时任务是灵魂**
```

为什么？

- 让系统"推着你走"
- 不依赖"想起来了才用"
- 形成稳定的工作节奏

案例：

- 每天9点推送选题
- 每天23点生成日志
- 每周一生成周报

```
**铁律3：记录一切**
```

为什么？

- 数据驱动决策
- 避免重复劳动
- 持续优化系统

记录什么？

- 每天做了什么
- 用了什么工具
- 遇到什么问题

- 效果如何

工具：

- Notion（结构化数据）
- 日志文件（原始数据）
- 截图（视觉记录）

```
**铁律4：快速迭代**
```

不要追求完美：

- 先跑通最小闭环
- 快速上线测试
- 根据反馈迭代

案例1：

- 第一版风格文件：500字
- 持续优化到3000字

案例2：

- 第一版海报：转化率5%
- 第二版海报：转化率20%

```
**铁律5：专注核心价值**
```

什么是核心价值？

- 判断（选题、质量、方向）
- 不是执行（写作、设计、发布）

把重复的交给系统
把判断留给自己

案例1：

- Agent负责：搜索、写作、发布
- 人负责：选题、审核、优化

案例2：

- AI负责：海报、系统、文案
- 人负责：定价、策略、运营

```
### 15.3.5 一人公司的成本结构

**案例1成本分析**：

| 成本项 | 月费用 | 说明 |
|--------|--------|------|
| OpenClaw | ¥0 | 开源免费 |
| API费用 | ¥50-200 | 根据使用量 |
| 飞书 | ¥0 | 免费版够用 |
| 字流 | ¥99 | 月付 |
| 服务器 | ¥20 | 腾讯云轻量 |
| **总计** | **¥169-319** | **月成本** |

**收益**：
- 节省时间：每周20小时
- 时薪¥200计算：每月节省¥16,000
- ROI：50倍+

**案例2成本分析**：

| 成本项 | 费用 | 说明 |
|--------|------|------|
| OpenClaw | ¥0 | 开源免费 |
| API费用 | ¥100 | 5个Bot |
| Notion | ¥0 | 免费版 |
| 额度成本 | $5,000 | 自有平台，边际成本低 |
| 推广费用 | ¥0 | 自然流量 |
| **总计** | **¥100 + $5,000** | **一次性成本** |

**收益**：
- 红包收入：¥4,900
- 后续API消费：持续收入
- 用户LTV：¥200-500/人
- ROI：5-10倍

### 15.3.6 一人公司的增长飞轮

**案例1的飞轮**：
```

内容创作
↓
全平台发布
↓
粉丝增长
↓
影响力提升
↓
更多机会
↓
更多内容素材

↓
（循环）

```
**案例2的飞轮**：
```

工具产品
↓
内容教程
↓
付费社群
↓
用户使用工具
↓
口碑传播
↓
更多用户
↓
（循环）

```
**核心要素**：
```

1. 找到你的核心能力
   - 案例1：内容创作
   - 案例2：技术产品
2. 用AI放大这个能力
   - 案例1：10倍内容产出
   - 案例2：1天搭建系统
3. 建立增长飞轮
   - 案例1：内容→粉丝→影响力
   - 案例2：工具→内容→社群
4. 持续优化系统
   - 数据驱动
   - 快速迭代
   - 不断提升

```
---

## 15.5 可复制的自动化模板

### 15.4.1 内容创作自动化模板

**1. 目录结构**：
```

```
~/.openclaw/
├── workspace/
│   ├── SOUL.md # Agent角色定义
│   ├── writing-style.md # 写作风格规范
│   └── memory/ # 记忆系统
│   ├── 2026-02-11.md
│   ├── 2026-02-12.md
│   └── ...
├── scripts/
│   ├── daily-topic-push.sh # 每日选题推送
│   ├── generate-article.sh # 生成文章
│   └── publish-all.sh # 发布到所有平台
└── config/
├── feishu.json # 飞书配置
├── ziliu.json # 字流配置
└── platforms.json # 平台配置
```

2. SOUL.md模板：

```markdown
# Agent角色定义

## 身份
你是"[Agent名称]"，一个专业的[职责]助手。

## 核心职责
1. [职责1]
2. [职责2]
3. [职责3]

## 工作风格
- [风格要求1]
- [风格要求2]
- [风格要求3]

## 工具权限
- 可以调用：[工具列表]
- 不可以：[限制列表]

## 工作流程
1. [步骤1]
2. [步骤2]
3. [步骤3]

## 边界
- [边界1]
- [边界2]
- [边界3]
```text
**3. 定时任务模板**：

```bash
# crontab -e

# 每日选题推送（早上9点）
0 9 * * * /path/to/openclaw run daily-topic-push

# 每日工作日志（晚上11点）
0 23 * * * /path/to/openclaw run daily-summary

# 每周周报（周一早上8点）
0 8 * * 1 /path/to/openclaw run weekly-report

# 每月月报（每月1号早上9点）
0 9 1 * * /path/to/openclaw run monthly-report
```text
**4. 发布脚本模板**：

```bash
#!/bin/bash
# publish-all.sh

# 1. 从飞书获取文章
article=$(openclaw feishu get-article "$1")

# 2. 推送到字流
draft_id=$(openclaw ziliu create-draft "$article")

# 3. 发布到所有平台
openclaw ziliu publish "$draft_id" \
  --platforms
"zhihu,juejin,bilibili,xiaohongshu,wechat,csdn,segmentfault,jianshu,toutiao,baidu,sohu,163,sina,qq"

# 4. 记录发布日志
echo "$(date): Published article $1" >> /var/log/openclaw/publish.log
```text
### 15.4.2 社群运营自动化模板

**1. 目录结构**：
```

```
~/.openclaw/
├── bots/
│   ├── personal-assistant/ # 私人助理Bot
│   ├── content-bot/ # 内容Bot
│   ├── learning-bot/ # 学习Bot
│   ├── team-bot/ # 团队Bot
│   └── outreach-bot/ # 出海Bot
├── automation/
│   ├── payment.py # 支付自动化
│   ├── credit.py # 额度管理
│   └── watchdog.py # 模型看门狗
└── data/
    ├── users.csv # 用户数据
    ├── credits.csv # 额度记录
    └── logs/ # 日志文件
```
```

2. Bot配置模板：

```markdown
# Bot配置

## 角色
你是[Bot名称]，专门负责[职责]。

## 职责
1. [职责1]
2. [职责2]
3. [职责3]

## 工具权限
- 可以调用：[工具列表]
- 不可以：[限制列表]

## 工作流程
1. [步骤1]
2. [步骤2]
3. [步骤3]

## 输出格式
[格式说明]
```text
**3. 支付自动化模板**：

```python
# payment_automation.py

from flask import Flask, request
import qrcode

app = Flask(__name__)

@app.route('/pay', methods=['POST'])
def handle_payment():
    # 1. 接收付款通知
    payment_data = request.json

    # 2. 验证付款
    if verify_payment(payment_data):
        # 3. 生成群二维码
        qr_code = generate_group_qrcode()

        # 4. 发送给用户
        send_qrcode_to_user(
            user_id=payment_data['user_id'],
            qr_code=qr_code
        )
```

```
        # 5. 记录到数据库
        save_to_database(payment_data)

        return {"status": "success"}

    return {"status": "failed"}

if __name__ == '__main__':
    app.run(port=5000)
```text
**4. 额度管理模板**：

```python
# credit_management.py

import pandas as pd
from datetime import datetime

class CreditManager:
    def __init__(self, db_path='data/credits.csv'):
        self.db_path = db_path
        self.load_database()

    def load_database(self):
        """加载数据库"""
        try:
            self.db = pd.read_csv(self.db_path)
        except:
            self.db = pd.DataFrame(columns=[
                'user_id', 'wechat', 'amount', 'status', 'created_at'
            ])

    def add_user(self, user_id, wechat, amount=50):
        """添加用户"""
        self.db = self.db.append({
            'user_id': user_id,
            'wechat': wechat,
            'amount': amount,
            'status': 'pending',
            'created_at': datetime.now()
        }, ignore_index=True)
        self.save_database()

    def distribute_credit(self, user_id):
        """发放额度"""
        if self.is_distributed(user_id):
            return {"error": "已发放"}

        # 调用API发放
        result = api_distribute_credit(user_id, amount=50)

        # 更新状态
```

```
        self.db.loc[self.db['user_id'] == user_id, 'status'] = 'distributed'
        self.save_database()

        return result

    def is_distributed(self, user_id):
        """检查是否已发放"""
        row = self.db[self.db['user_id'] == user_id]
        return len(row) > 0 and row.iloc[0]['status'] == 'distributed'

    def save_database(self):
        """保存数据库"""
        self.db.to_csv(self.db_path, index=False)

    def export_report(self):
        """导出报告"""
        return self.db.to_csv('credit_report.csv')
```text
### 15.4.3 快速开始指南

**步骤1：选择场景**
```

场景A：内容创作自动化

- 适合：自媒体、博主、内容创作者
- 核心：定时选题 + 自动生成 + 多平台发布
- 效果：3小时 → 10分钟

场景B：社群运营自动化

- 适合：创业者、产品经理、社群运营
- 核心：自动化付款 + 额度管理 + 多Bot矩阵
- 效果：1天冷启动100人

场景C：混合场景

- 结合A和B
- 内容+社群双轮驱动
```

步骤2：搭建基础设施

```bash
# 1. 安装 OpenClaw
curl -fsSL https://openclaw.example/install.sh | bash  # 替换为实际安装地址

# 2. 创建工作目录
mkdir -p ~/.openclaw/workspace
mkdir -p ~/.openclaw/scripts
mkdir -p ~/.openclaw/config

# 3. 复制模板文件
cp templates/SOUL.md ~/.openclaw/workspace/
cp templates/writing-style.md ~/.openclaw/workspace/
cp templates/*.sh ~/.openclaw/scripts/

# 4. 配置API
openclaw config set api.key "your-api-key"
openclaw config set model "claude-opus-4"
```text
**步骤3：配置定时任务**

```bash
# 编辑crontab
crontab -e

# 添加定时任务
0 9 * * * /path/to/openclaw run daily-topic-push
0 23 * * * /path/to/openclaw run daily-summary
```text
**步骤4：测试运行**

```bash
# 测试选题推送
openclaw run daily-topic-push

# 测试文章生成
openclaw ask "写一篇关于AI编程的文章"

# 测试发布
openclaw run publish-all "article-id"
```text
**步骤5：持续优化**

1. 收集数据
   - 记录每次运行的结果
   - 分析效果数据
2. 优化配置

- 调整风格文件
- 优化提示词
- 改进流程

3. 迭代升级

- 根据反馈改进
- 添加新功能
- 提升自动化程度

---

## 📝 本章小结

通过2个真实的一人公司案例，学习了：

### 核心收获

**案例1：内容创作自动化**
- 效率提升：3小时 → 10分钟（94%）
- 关键系统：定时任务 + 风格文件 + 记忆系统
- 核心价值：把重复的交给系统，把判断留给自己

**案例2：社群运营自动化**
- 冷启动：1天100人，收入¥4,900+
- 关键系统：支付自动化 + 多Bot矩阵 + 模型看门狗
- 核心价值：用AI放大核心能力，建立增长飞轮

### 方法论总结

**一人公司的3个核心能力**：
1. 判断力（AI负责执行，人负责判断）
2. 系统化思维（搭建AI系统，而不是用AI做事）
3. 快速迭代能力（先跑通最小闭环，再持续优化）

**一人公司的4个关键系统**：
1. 内容生产系统
2. 社群运营系统
3. 模型管理系统
4. 数据记录系统

**一人公司的5个铁律**：
1. 绝不自动发布
2. 定时任务是灵魂
3. 记录一切
4. 快速迭代
5. 专注核心价值

### 实战启示

**AI真的颠覆了生产力**：
- 不是内容本身被AI替代
- 而是内容生产的效率被AI拉高了一个数量级
- 以前需要一个团队干的活，现在一个人加上AI就能搞定

**一人公司的未来**：
- 更多人会选择一人公司模式
- AI是最好的"员工"
- 核心竞争力是判断力和系统化思维

### 下一步行动

1. **选择场景**：内容创作 or 社群运营 or 混合
2. **搭建系统**：按照模板快速搭建
3. **测试运行**：跑通最小闭环
4. **持续优化**：数据驱动迭代

开始你的一人公司之旅，用AI放大你的核心能力！

---

**全书完**

恭喜你完成了《OpenClaw完全指南》的学习！

从基础入门到高级应用，从技术配置到商业实战，你已经掌握了OpenClaw的完整知识体系。

现在，是时候开始你自己的AI自动化之旅了！

💡 **记住**：
- AI负责效率，人负责判断
- 把重复的交给系统，把判断留给自己
- 快速行动，持续迭代

🚀 **祝你成功**！